

No part of the candidate's evidence in this exemplar material may be presented in an external assessment for the purpose of gaining an NZQA qualification or award.



Mana Tohu Mātauranga o Aotearoa
New Zealand Qualifications Authority

Scholarship 2023

Technology 93601

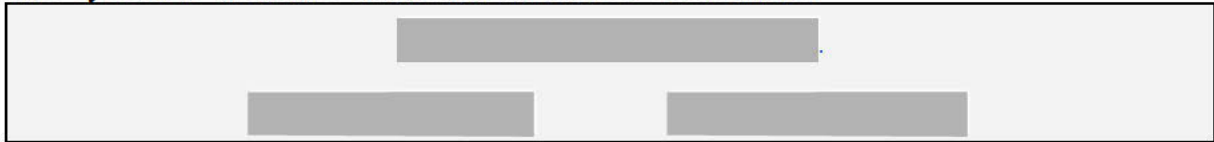
SCHOLARSHIP EXEMPLAR

Introduction

This document outlines my [REDACTED] journey in creating a database system for my client, a charity called Delta. This system manages, records and reports statistics of food parcels so Delta can more efficiently aid citizens in need. Delta does an incredible job backed by a wonderful team of volunteers and it has been my pleasure to add to their charity using my skills and resources.

System credentials

The system is available at this link with these credentials.



Identified problem

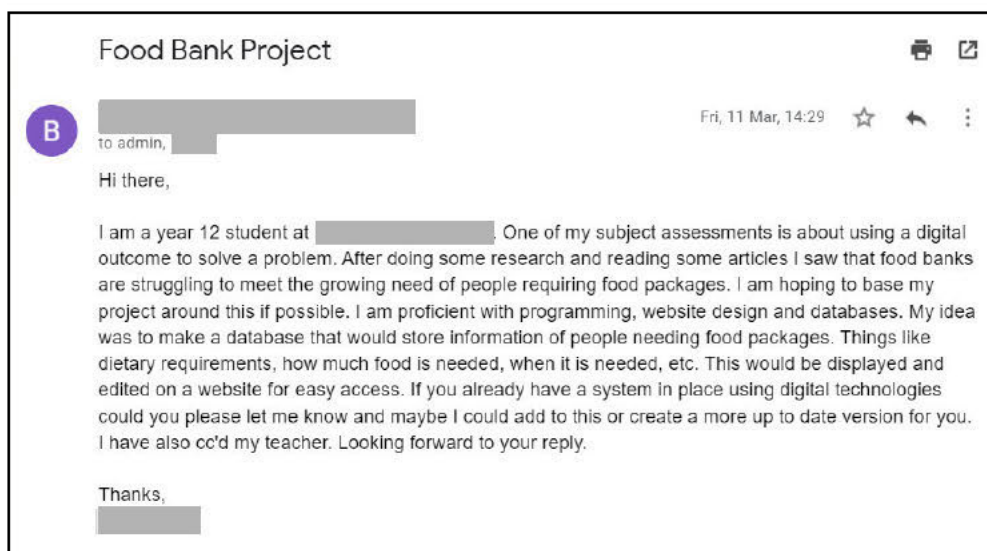
Many people in New Zealand are finding themselves struggling to meet their family needs. [REDACTED]

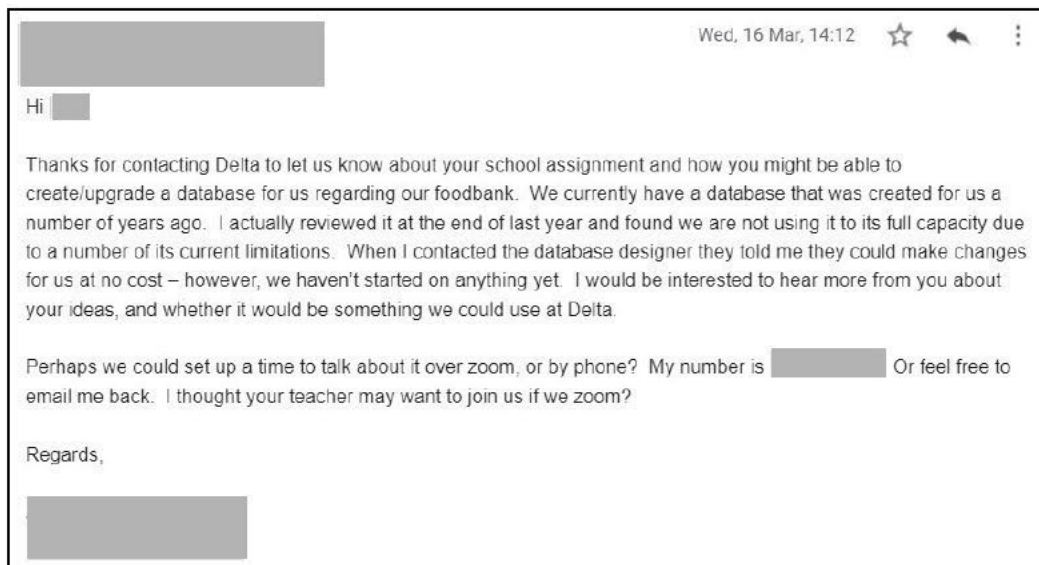
” - [Stuff News article](#)

This is a huge issue and got me wondering, how can I use my digital skillset to give aid or help those giving aid

Finding client

I explored food banks within my local community and came up with several local food banks which I then emailed to find out how I could help them using Digital Technologies. Some replied saying that they had adequate software but one Food Bank called Delta Trust replied specifying that they require a database for their record keeping.





Zoom meeting with stakeholder

I set up a Zoom meeting with Delta and discussed what they would require. Their current system does not record enough information from clients and doesn't provide the reporting they need, which is mainly statistics for their funders around demographics. Funders need to know how their money is being spent. There are also security concerns with the current system. Delta provides services in addition to the food bank, such as programmes for elderly people and intellectually disabled adults. Their current system does not cover these services. Delta wants a database that can support all of these programmes, not just the foodbank. [redacted] from Delta agreed to write up a document outlining their requirements.

Delta's brief

The following document was provided by [redacted], and covers the current system problems and requirements.

DATABASE INFORMATION FOR [redacted]

By [redacted]

You contacted us with a desire to create a database for a Foodbank. Whilst a foodbank is a large part of what we do, at [redacted] we have three different departments; Community Development Services (CDS) which runs our Foodbank, community meal and café, and cultural programs for refugees and migrants including ESOL classes and an international playgroup. Our Evergreen Club runs a day programme for over 65's, where they come to socialise, engage in activities, do some light exercise and have a hot meal and morning tea. Our Friendship Link is a day programme for intellectually disabled adults who join in activities such as craft, karaoke, dancing, discos and movie nights, and outings. You can get more information about what Delta does from our website <http://www.deltatrust.org.nz>

We currently have a database system, however, it is so far designed as such that we cannot retrieve any information from it. We can look up info on screen, but there is no way for us to print or download any reports. Other than that, it is still lacking in a number of other functions, and so we don't use a database to the capacity that we would like. It is for this reason Delta welcomes your voluntary time and expertise to create a new, user-friendly, and fit-for-purpose database for us.

Our requirements

█ is a Charitable Trust and is required to report to our Funders how we have spent their money, and what we have achieved with the resources we have been given. For this reason (and also for our own internal knowledge) we need to track who we are helping, how we are helping them, and what outcomes we are achieving. A well-designed database will help us to achieve this.

Functionality

- Each client's record needs to have contact details, date of birth, ethnicity, gender, number in household and children's dates of birth, income source, dietary requirements, how they heard about us etc. Possibility to link their record to a partner who is also on our database (so we don't double-up with food parcels)
- Each client's record to be either 'Active' or 'Inactive'
- Ability to assign each client to one or more of the three departments
- Ability to assign each client to a subcategory of a department (ie Foodbank, or budget advice and advocacy) and to be able to record each instance of contact with us/activity log/ appointment notes. ie if they came in for a food parcel on a Monday and then had a session with our budget advisor on the Wednesday, and were then referred to a government department for housing etc. Each email, phone call or activity log needs to be under a particular sub-category group so we can record them to that service, and allow us to report from that.
- Each sub-category (ie Programme) will need to have slightly different additional information that needs to be recorded.

Reporting

- Since each funder gives us money for a set period of time (usually 12 months, but could be 3 months or 6 months) and in various months throughout the year, we need flexibility in our reporting on which month we start from and therefore need to be able to set the 'date from and to' parameters.
- From each service program we will need to be able to report on the individual sessions, as well as a report for total instances of support i.e. phone calls + emails + attended sessions.
- For the Foodbank we need to report on how many food parcels we have given out and whether they were individual parcels or family parcels. And probably how many times people are using the service in the last 12 months etc
- We need to report on each staff member running the programs (i.e. we have more than one budget advisor and advocacy worker) so we can see if they are achieving their KPI's.

Secondary additional document

█ provided an additional document expanding on their requirements.

Issues with our current database

1. Lack of reporting functionality
2. Lack of input variables
3. Potential for being hacked
4. Not user friendly/ no ability to make major changes ourselves (ie to add in a new drop-down box etc)

Reporting:

- Basically, there is no reporting function, so we are unable to extract any data except in seeing an individual record on screen. This is extremely limiting for us when it comes to reporting back to our Funders, and indeed for us to assess whether our programmes are working as we intend. Therefore all our statistics are taken manually and entered into a spreadsheet. However, these are only basic stats and don't give us the full picture. Ie we currently count how many food parcels are given in a year, and of those, how many were family parcels or individual parcels. But this does not tell us things like how many people were in each family (and therefore how many people did we actually feed), what percentage of those recipients were Māori, or Pasifika, or MELAA. Or what ages are the people we are feeding.
- We have no option to print anything
- We have no way to search or print any collective data (ie number of
- We have no way to search over a set time period (ie 1 Sep 2020 – 31 Aug 2021) – and our time periods always change per Funder for which 12 months (or 3 months) we need to report on.
- We can't report on how many times one individual has been helped

Inputs:

- Our current database doesn't have 'drop down options' for things like Ethnicity, (currently it is a data entry field, and not consistent enough to search on) so we can't report on any collective data.
- We are not able to log individual sessions of help by different staff members, and for different programmes (though there is some ability in this regard)
- We don't have the ability to change the date of an entry if it wasn't initially entered on the date it happened, and has been entered later.

Hacking:

- The author of our current database recently commented that *"the type of data you have in my system presents a very high risk to me from the privacy law point of view if my servers are hacked."* He said we should look at the *"possibility of putting an application on your server."* ie MS ACCESS]

User experience:

- I have not had a lot of experience with using our current database, but it did not seem like it was very user friendly. We have a number of staff who are very hesitant to even work with it, and so it is not used in all departments – or indeed outside of the foodbank programme.
- We are not able to make any changes to the database ourselves. Ie adding in a new programme that we put on, or 'hiding' programmes that we don't currently use but still need to report on in the future. Additionally we are unable to add new drop-down boxes etc.
- We are unable to customise the 'look' of the database.

Breaking down the brief

In order to understand what the stakeholder wants I needed to break down the Brief into individual goals and tasks. This then enabled me to ensure that I was ready for the development of the Prototype Database.

Project goals:

- Creating a database
- Presenting that database with a website where it is editable, searchable and sortable
- Deliver a sustainable system that is future proof so that [REDACTED] can use it long term

Tasks:

- Creating a server
- Registering a domain
- Make a web host server
- Create a database
- Create a front end website to edit, search and sort the database
- Connect the website and database with SQL and Python

Requirements

- [REDACTED] needs a new well designed database that stores information about clients
- The database needs to be editable and sortable
- [REDACTED] need to be able to add new clients
- [REDACTED] need to be able to add new departments, subcategories and new personal information fields
- The database needs to be able to track who [REDACTED] are helping, how they are helping and what outcomes they are achieving.

Specifications**Functionality:**

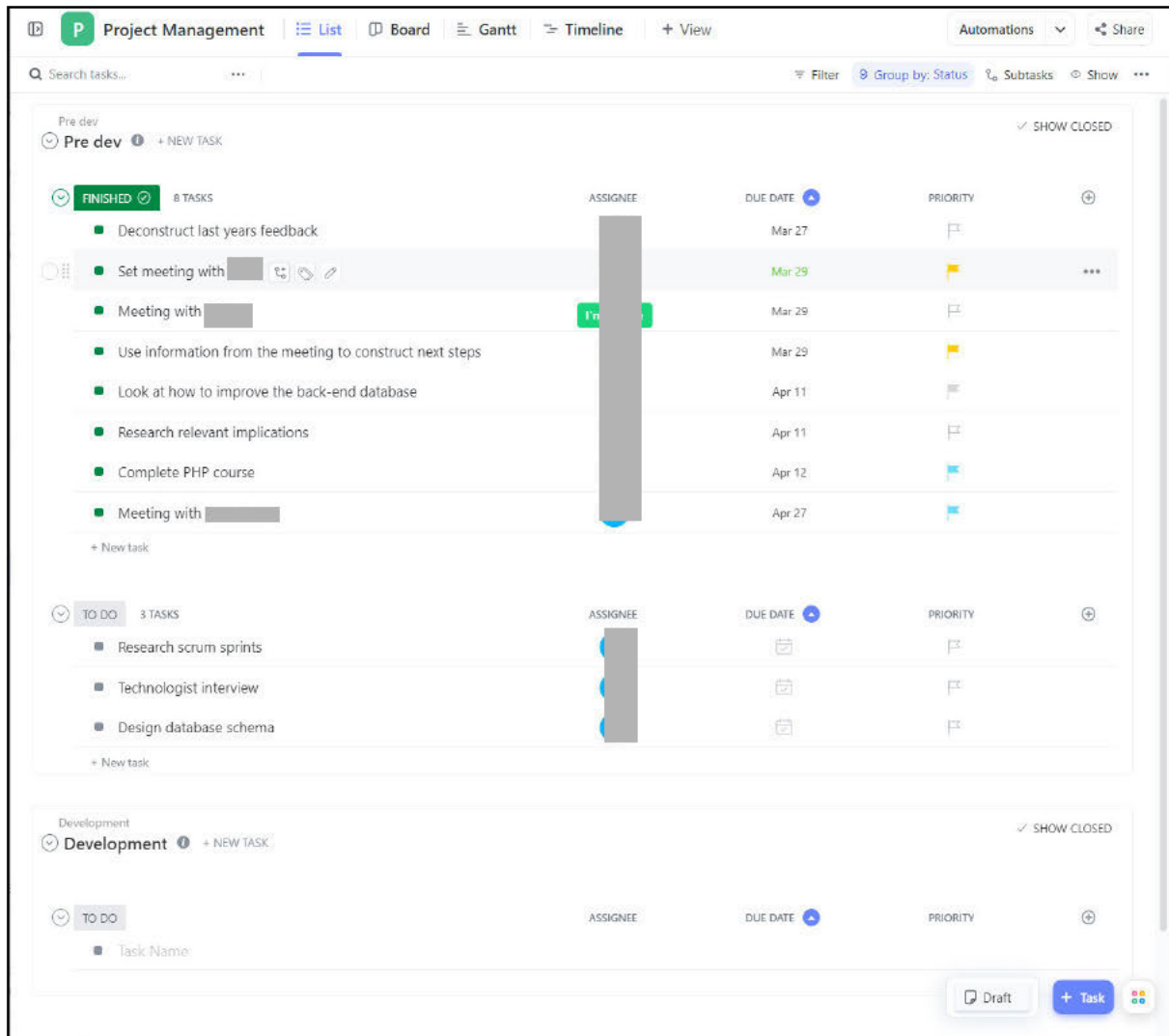
- Each client's record needs to have contact details, date of birth, ethnicity, gender, number in household and children's dates of birth, income source, dietary requirements and how they heard about [REDACTED]
- Clients also need to have links to their partners and family
- Each client's record to be either 'Active' or 'Inactive'
- Ability to assign each client to one or more of the three departments
- Ability to assign each client to a subcategory of a department
- Each sub-category needs to have different additional information that needs to be recorded

Reporting:

- Reporting on who is being helped and how many people are being helped and over specific periods of time
- Each service program needs to be able to report on the individual sessions, as well as a report for total instances of support i.e. phone calls, emails, attended sessions. This needs to be linked to the profile of the client involved
- Reporting on each department, for example for the foodbank [REDACTED], needs to know how many food parcels are going out and whether they are individual parcels or family parcels and how many people are receiving food parcels over specific periods of time.
- Reports need to be printable
- Feature to search for specific data, for example age or ethnicity
- Reporting on how many times one individual or a family has been helped
- Reporting on each staff member running the programs

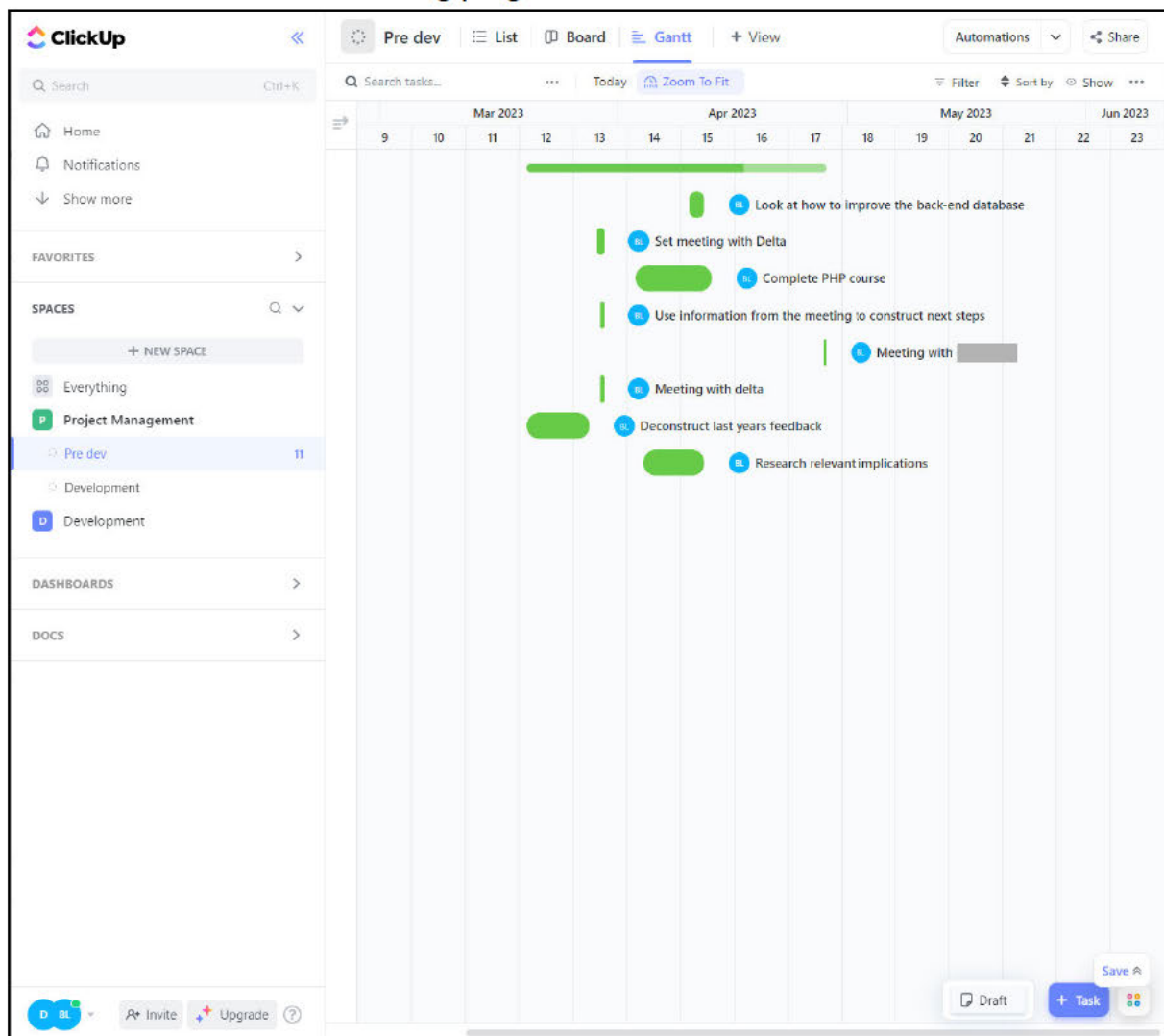
Project management tool

After searching for project management tools I came across an online application called ClickUp which looked perfect. ClickUp offers a centralised platform for project management where you can add tasks and set timeframes. The user interface is great, it has advanced features and is on the complex side but this is a long term project so it's good to be fully enhanced and customisable. Below is a screenshot of the main page where tasks can be added.



The second image is an example of the many benefits of this application. I can add multiple views such as a Gantt chart which lets me visualise my progress over time. This helps with staying on track and motivation because I can see my completed

tasks and know that I am making progress.



Database research

I researched several database design techniques before settling on a relational database model with MySQL as the database engine. The techniques I considered were Entity-Attribute-Value (EAV), Relational, Document, and Graph.

Entity-Attribute-Value (EAV) Model

How it works:

EAV is a highly flexible model where new features can be added without affecting the database structure. Instead of having a dedicated table for each type of record, all data is held in three tables. One representing the entity, one for each property type, and one for the value of each property for each entity.

Pros:

High degree of flexibility, and suitable for adapting to changes over time.

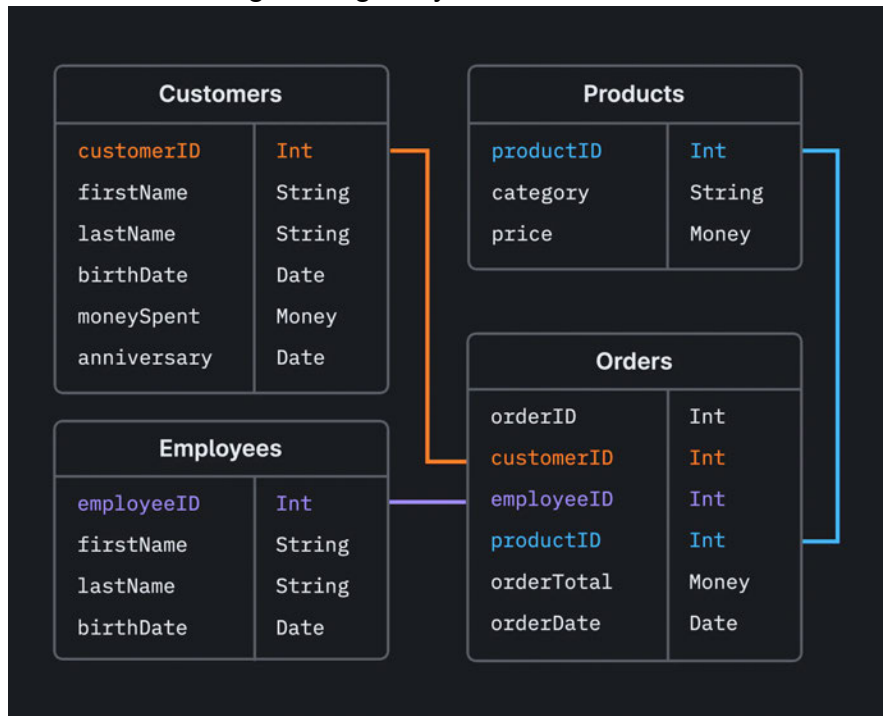
Cons:

Significant development time is needed, and it would add a lot of complexity to the code, which is not needed for the scale of the project yet. For now it makes more sense to focus on [REDACTED]'s immediate requirements rather than an EAV database.

Relational Model

How it works:

The Relational Model organises data into structured tables, each with predefined columns representing attributes. Entities are stored as rows, and relationships are established through foreign keys that link to other tables.:

**Pros:**

The relational model is convenient to use with SQL statements, and a popular choice for data capture and reporting systems like [REDACTED]'s requirements.

Cons:

Every change requires updating the database schema, for example adding a meal preference would need a corresponding column in the Person table.

Document Model

How it works:

In the Document Model, data is stored in semi-structured or unstructured documents, typically using formats like JSON or XML, as in this example:

```
{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": " ",
  "lastname": " ",
  "address": {
    "street": " ",
    "city": " ",
    "state": " ",
    "zip": " "
  }
  "hobbies": ["surfing", "coding"]
}
```

Pros:

The document model is suitable for unstructured or semi-structured data.

Cons:

Lack of querying language, difficulty of bring together many records for reporting. Due to being unstructured, maintaining data integrity can pose challenges.

Graph Model**How it works:**

The Graph Model represents data as nodes (entities) and edges (relationships) in a graph structure. Each node contains properties, and edges define connections between nodes.

Pros:

The graph model excels in representing and traversing complex relationships between data entities. This makes it suitable for scenarios where understanding connections and relationships is crucial.

Cons:

Increased storage requirements and complexity in implementation can pose challenges.

Conclusion

Considering the unique requirements of Delta, my research has shown that a Relational Model is the most suitable option. The structured nature aligns with the foodbank's data types and relationships, and ease of use make it the best choice. While other models would work, they introduce a level of complexity that isn't needed to meet Delta's requirements.

Bibliography

1. <https://planetscale.com/blog/schema-design-101-relational-databases>
2. <https://www.mongodb.com/document-databases>

Consulting technologist

I interviewed a family friend who is a programmer and has created similar systems to mine in the past. In the interview, I asked him the following questions

Question 1: What are your thoughts on my choice of database model?

Answer: “The data model you’ve chosen is a very traditional and industry standard way of saving application data. The database tables closely match the UI and there is a predictable relationship between the UI and underlying data. Future updates and features will be easy to implement with this approach. The pattern is to add additional tables and columns on an as-needed basis.

An alternative data model would be an Entity/Attribute/Value (EAV) model where the objects being saved don’t have dedicated tables in the database. There are pros and cons, and in this case I would not recommend an EAV model due to the increased complexity without corresponding benefits for a small to medium project of this nature.

Question 2: What programming language would you recommend?

Answer: “PHP would be the ideal choice for the backend. It’s free, widely used, runs on different platforms and is well documented”

Question 3: Are there any tools that you can think of that I can utilise for my system?

Answer: “I recommend you look into Materialise CSS and HighCharts. I've done a lot research on tools that help web development and I think these two would be great to use

Question 4: My client wants an address auto-complete feature, is there a tool I can use to help me?

Answer: I’ve heard good things about Google’s address API. It’s the same thing that powers Google maps and I believe it is open source.

Relevant implications

To ensure my thinking is broad and that I consider all aspects I decided to list some relevant implications. This will help me objectively think about the project which in turn will help me deliver a more coherent functioning product that ticks all of the boxes.

Privacy	<p>Confidentiality Names of families receiving food parcels need to be strictly confidential, and therefore robust security measures are needed to limit access to the data.</p> <p>Data Privacy and Security Because the database will contain personal information of food bank recipients, elderly club members, and mentally disabled adults, I need to ensure that the software complies with data privacy regulations like the NZ Privacy Act.</p> <p>Data Collection To reduce risk I will limit the data being collected to only what [REDACTED] requires.</p>
Intellectual	Ownership and Copyright

property	<p>Since I'm creating this database system as part of my school project for [REDACTED], it's crucial to establish who owns the rights to the software.</p> <p>Third party software I will most likely use existing technologies, languages and libraries to develop the database system. I need to be aware of any licensing requirements for these components.</p>
End-user considerations	<p>End users The end users are [REDACTED] employees. They will download reports and send them to funders.</p> <p>[REDACTED] clients People receiving food parcels won't have access to the system.</p> <p>[REDACTED] One of [REDACTED]'s main concerns is to secure future funding so they can better help people.</p> <p>[REDACTED] funders What's important for funders is that they want to know that they have made a difference. It's important for them to know exactly what impact their donations have had on struggling New Zealanders. Suitable reporting of trends over time is important.</p>
Cultural and Ethical	<p>Diverse Distribution of Food Parcels Ensuring a fair and diverse distribution process of the food parcels is very important. The database system will track demographic information and make this available in reports.</p> <p>Transparent Data Collection Clients need to be aware their data is being collected, including food parcel history.</p>
Design considerations	<p>Usability Making the system simple and easy to use is important, so that minimal training is needed for [REDACTED] employees.</p> <p>Integration [REDACTED]'s client list and food parcel history needs to be imported into the database to enable historic reporting. This data needs to be extracted from their current system.</p> <p>User Interface (UI) Design I want to ensure that the system is modern and sleek looking, and a pleasure for [REDACTED] employees to use. It should be very intuitive too. To achieve this, I'll consider elements like colour schemes, typography, and layout. Incorporating [REDACTED]'s branding colours and design elements will help establish a strong connection to the</p>

	<p>system. Icons, buttons, and navigation should be clear and consistent across all modules.</p> <p>Feedback After [REDACTED] begins using my system I will ask for feedback on the aesthetics of the program and if there should be any changes, this allows for time for [REDACTED] to try the system out and form strong opinions. I still have feedback from last year's prototype to go off of.</p> <p>Device Compatibility The system will be used on web browsers running on PCs and laptops. I will make sure it is compatible with Chrome, FireFox, Edge and Safari. There is no immediate need for the system to be compatible with mobile browsers.</p>
Sustainability	<p>Long-Term Sustainability From my perspective, this is a long term project where I will be working with [REDACTED] for many years. Making good design decisions now will help in the future as the system grows with new features.</p> <p>Handling Glitches and Technical Issues Issue Resolution Plan: I will provide my contact information for technical assistance. This will be located on the contact page of the system.</p> <p>Regular Backups I will implement either automated or manual regular data backups to safeguard against data loss in the event of a major glitch or system failure.</p>
<p>Doing these implications made me realise certain aspects such as device compatibility, data privacy and third party software which I would not have considered beforehand. Now I am more informed and objective minded about the project I am ready to move onto the next step of planning conceptual designs of the database.</p>	

In person meeting with [REDACTED]

I emailed [REDACTED] and asked to set up an in person meeting with them. The purpose of this was to have a look at their current database and its problems but also to plan together the new database. I met with [REDACTED] and [REDACTED] from their administrative team. We discussed what [REDACTED]'s dream system would look like to them and how I can meet their needs. Together we sketched some wireframes of what different pages of the front end web page will look like, this is discussed further down in the document under the "Front end design" heading. Below are the written notes I took from that meeting, the notes are listing individual changes or add-ons that they want and the headings represent each tab I have on the website:

Home page

- 4 boxes of each department with information below each one showing statistics like instances of support of that department eg: Evergreen club and friendship link, sessions attended (number),

- Extra box at the bottom for total instances of support
- How many people come to cafe for CDS reporting
- Clicking on a department shows extra information of that department and also shows where to take the roll for things like evergreen club and friendship link

People Tab

- Extra box after ethnicity eg: European then European-NZ
- Calendar for date of birth so you can click on it
- DOB might need to have an unknown data value as clients might not know DOB
- Extra box for people showing what department they use or are linked to
- Extra table in people tab showing past activity and events
- Tallying up each number
- Support person coming in to pick up parcel for someone else and their contact details
- Adding parameters or filters
- With an extra box for community boards (reporting for where everyone is in community boards and percentage of people in each location).

Individual Person tab

- Auto-fill address when typing
- Dependencies or partners for each person: extra box where you can add dependencies with links to their pages. Children might not have extra links as no pages
- Advanced date picker
-

Adding activity in individual person tab

- Being able to refer people to other departments and show where they came from
- Portion sizes, single or family, how many in each family
- Activity - Partner received food parcel

Reporting

- Since each funder gives us money for a set period of time (usually 12 months, but could be 3 months or 6 months) and in various months throughout the year, we need flexibility in our reporting on which month we start from and therefore need to be able to set the 'date from and to' parameters.
- From each service program we will need to be able to report on the individual sessions, as well as a report for total instances of support i.e. phone calls + emails + attended sessions.
- For the Foodbank we need to report on how many food parcels we have given out and whether they were individual parcels or family parcels. And probably how many times people are using the service in the last 12 months etc
- We need to report on each staff member running the programs (i.e. we have more than one budget advisor and advocacy worker) so we can see if they are achieving their KPIs.

Foodbank

- 2 food parcel sizes: single parcel up to 2 people and family for more

Evergreen club

- Assigning people a day of the week that they came in. Each day staff would tick their name off on the roll. If they're not there then they need to tick a reason for that.

Friendship link

- Assigning people a day of the week that they come in. Each day staff would tick their name off on the roll. If they're not there then they need to tick a reason for that.

Other

- People call or email in to order food parcel

Extra database tables and fields

Roll taker, absentees reason

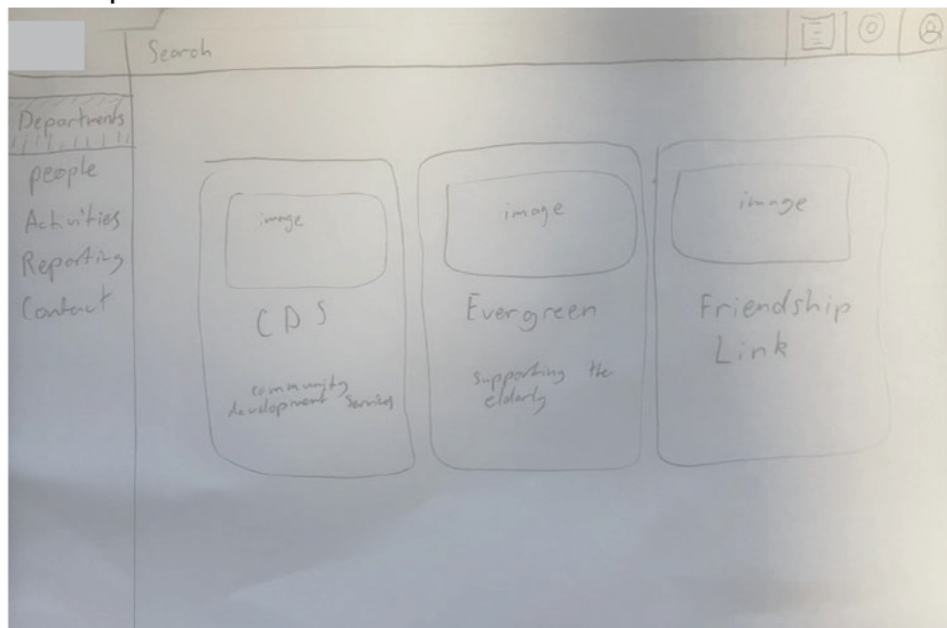
Links to dependencies, partners, children

Portion sizes

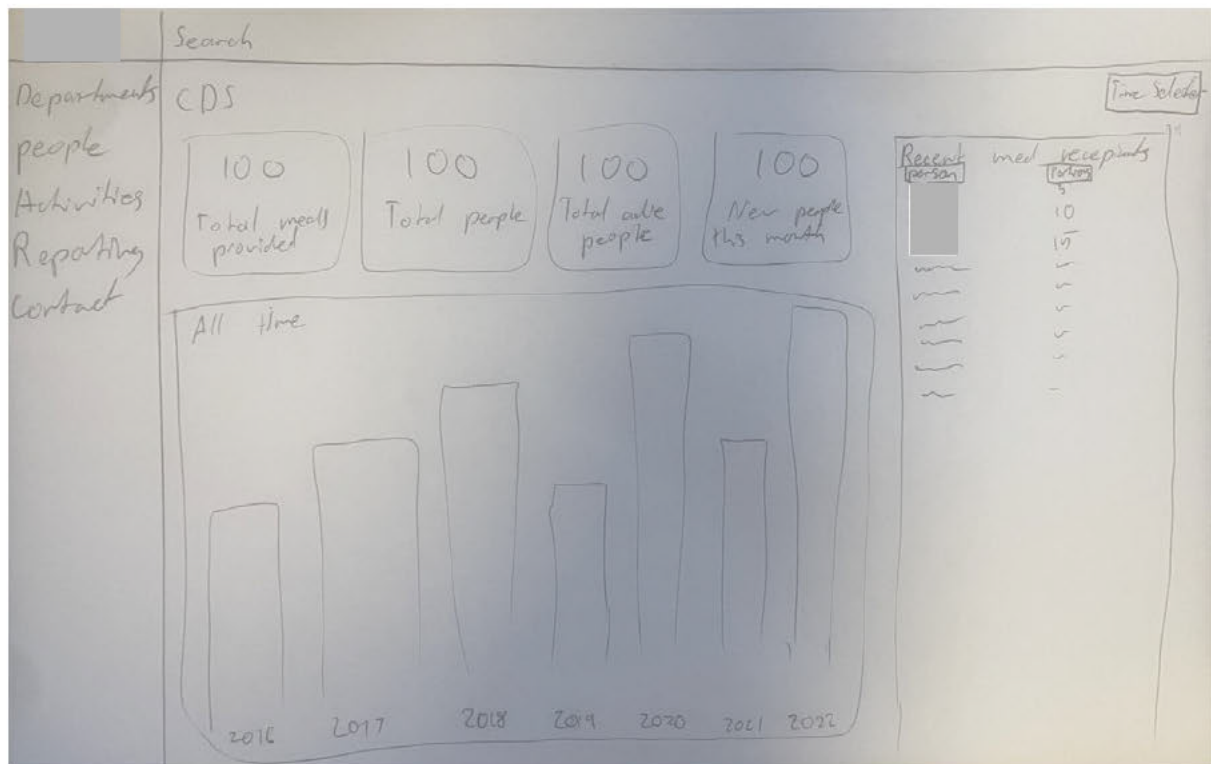
Front end conceptual designs

During the initial meeting with [REDACTED], I sketched some wireframes to help us share ideas. Despite being rough, they give a good idea of how the system should look. By working together, the stakeholder became part of the design which helped build excitement and develop our relationship.

There are around 20 different wireframes which is too many to show here. Below are 2 examples.



This sketch above is on the home/department page, the opening page when the user goes to [REDACTED]. Because of the different departments and certain staff only dealing with certain departments there needs to be a clear way of separating them. My client decided on having three different clickable cards for the departments to take the user to its respective dashboard.



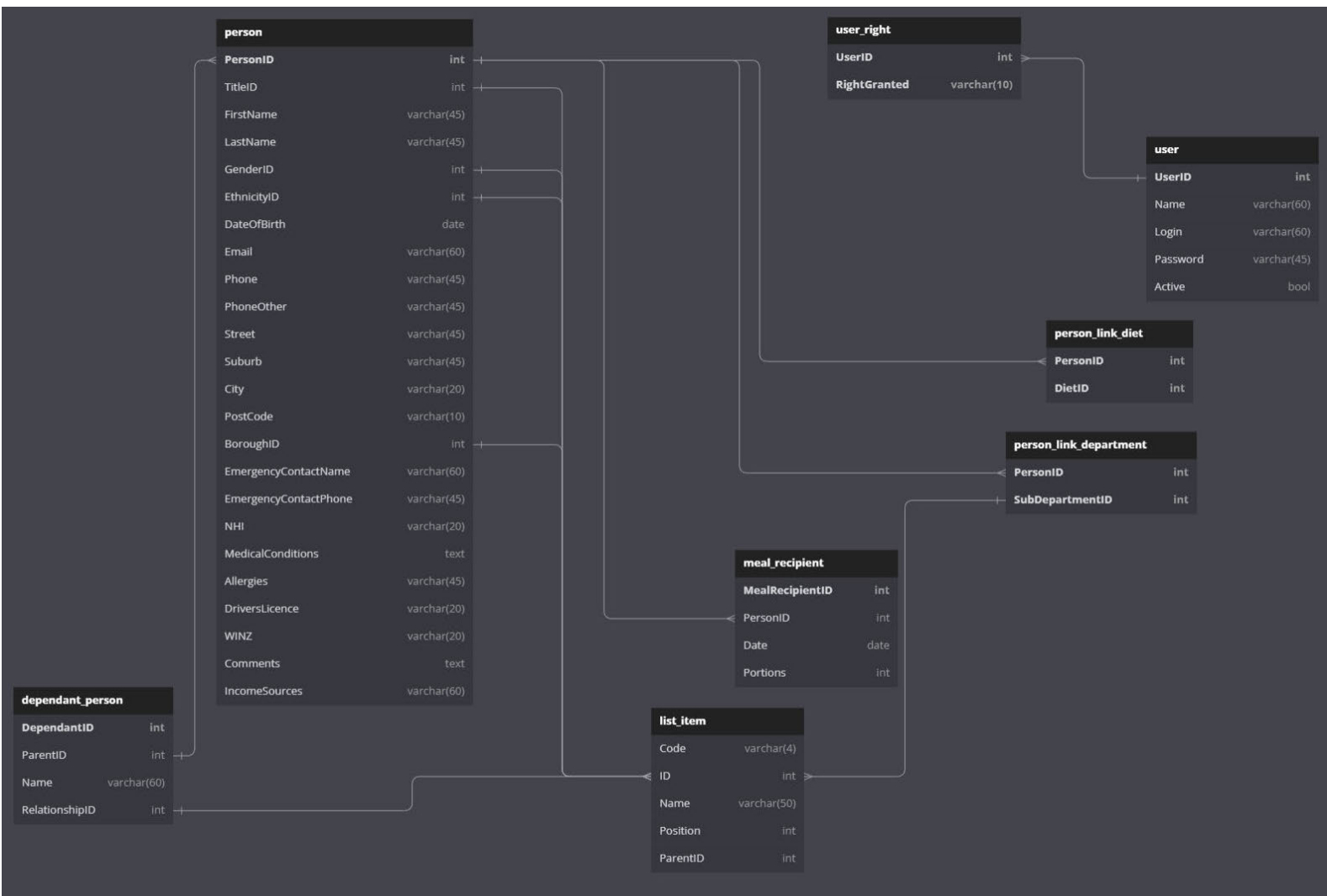
The second sketch is of the CDS (Community Development Services) dashboard, which is the food bank. At the top we decided there should be four boxes which show some general statistics such as total meals provided, total registered people, total active people and new people over a time period. The purpose of these boxes is to give more liveliness to the dashboard and a sense of accomplishment when staff see the impact they have had.

The list table on the right side displays the recent meal recipients. The bar graph on the bottom displays meals provided over a set amount of time. In the top right corner there is a button labelled time selector. When clicked this button will open a dropdown menu with different time selection options allowing the user to select a year or a month of a year.

Back end database conceptual designs

I have decided to plan and model my full database. This took a lot of time and thinking and may be subject to change in the future however I now have a better understanding of my relationships with my tables. This minimises the risk of making a mistake.

Below is a relational database plan using the modelling software "dbdiagram.io." Each table represents



Starting PHP course

I have decided to enrol myself in a [PHP course on Youtube](#). My reasoning for this is that I need to ensure my programming skills are up to par for this project. The last thing I want to do is to deliver a program with sloppy code that could give errors or be difficult to debug. It has also been some time since I have used PHP so there's a good chance that my skills will be rusty. My plan is to quickly complete the course before I move onto any development so I can bring my best to this project and better meet the requirements of my stakeholder. I believe that the cost of time spent doing this course will return greater results and is worth the investment.

I have now finished the PHP course. From it I have gained and relearned ideas and theory about PHP that will be vital when I reach the development stage. This will help me with building my prototype in that I will have confidence in building a database for a client. It will also ensure that the system will be more robust and efficient in my programming at the back end of the database. This also means that if any future changes need to be made the code is more coherent and easy to moderate.

Code structure

I've made comprehensive use of include files to structure my code. This lets me reuse functions and avoid duplicating large amounts of code. For example, the include file "inc-db.php" has a function to connect to the database called "connect_to_database". This is used by many other files, and means that there is only one place that connects to the database, and no duplication of code. I have other include files relating to the database, the HTML, forms, people, meals, the dashboard and so on. Together they provide good structure to the code and a clear pattern for future growth of the system.

Part 1 development / task list

Given the huge list of features and complexity I have been tasked with doing I am going to split the project up into two phases. The first phase is focused on basic functionality of being able to record people with their fields as well as being able to assign meals. The second phase

Phase 1

People page
Person view page
Person edit page
Add meal page

Phase 2

Address finder
Home page
Date picker
Reporting

Setting up server

After some research I have decided to use Amazon Web Services (AWS) to host my web server which makes my website accessible from any computer. AWS is widely considered the best web hoster for situations like mine. It's relatively cheap, reliably backed by one of the largest companies, has plenty of features and options, and has great protection against hacking which was one of [REDACTED]'s requirements. There is a cost of registering a domain name which is between \$10 - \$20 per year and also hosting a server which will cost \$12 - \$36. This totals to \$22 - \$56 per year which [REDACTED] are more than happy to pay for.

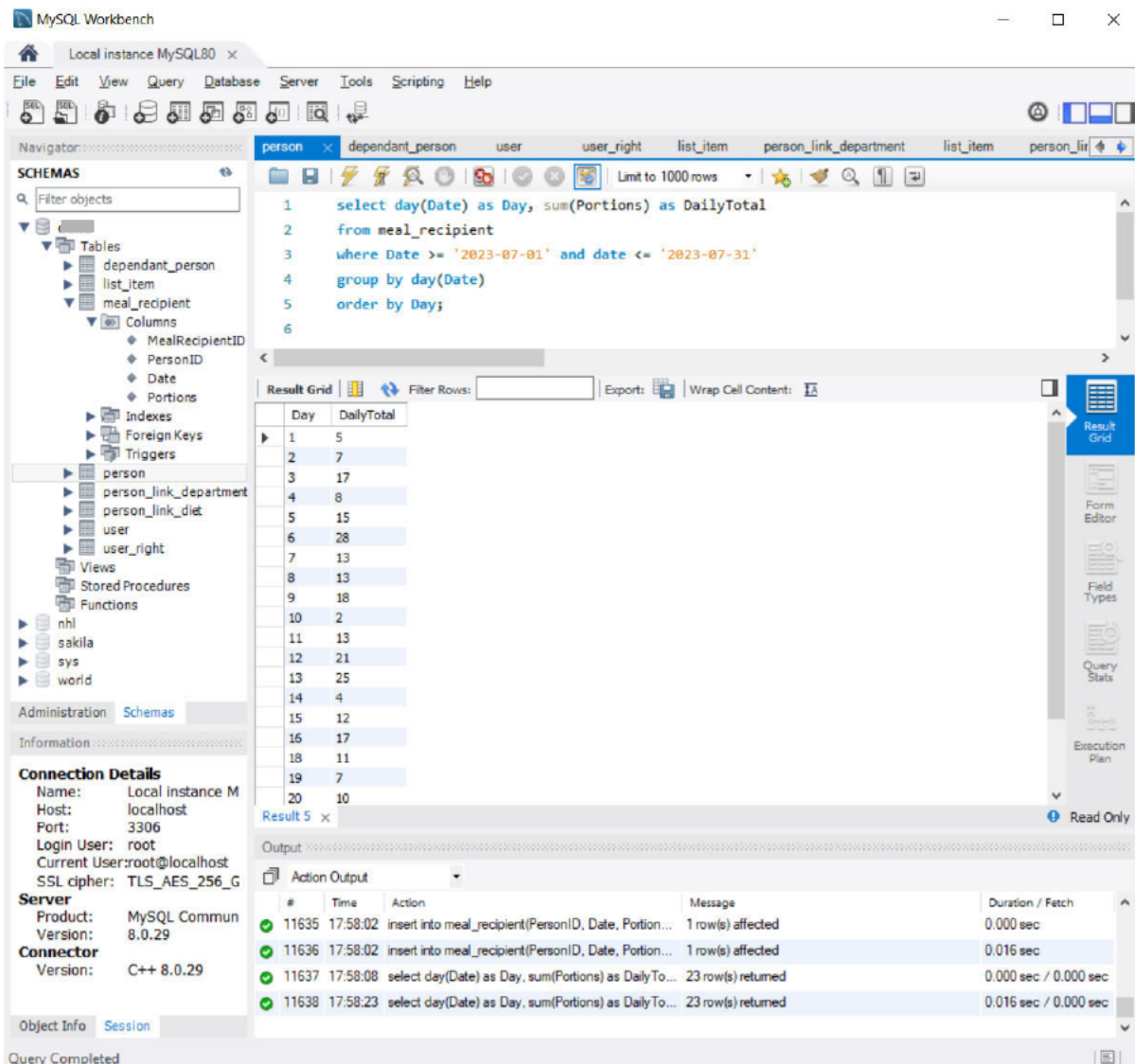
Data structure and tools

I have two separate databases, one on my home computer for development and one on my server for my client. I have decided to do this for time saving purposes. Writing SQL on my server means I have to use the Remote Desktop Protocol, and that interfacing with my server directly has lag and glitches. I write everything on my home computer and when I reach a milestone I go through the process of uploading it as a lump sum rather than uploading every small change I make. This saves uploading time and time in development as I can use my high end PC rather than AWS's slow computer.

I decided to use MySQL workbench as my archetype tool for my database. It would make no sense switching to something else and MySQL has caused no issues with me so I will continue to use it this year.

MySQL Workbench is a powerful, user-friendly database design and administration tool developed by Oracle for managing MySQL databases. It lets users design databases using entity relationship diagrams or ER diagrams which is exactly what I am doing. It writes and executes SQL queries with features like syntax highlighting

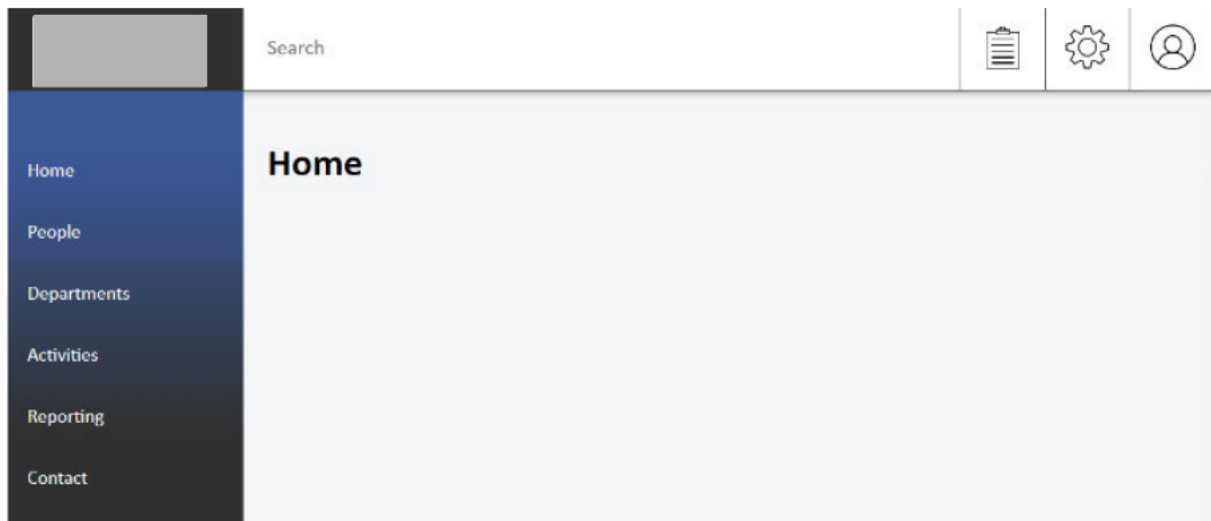
and code completion, monitors performance, and does backups. It's a very useful tool for MySQL database developers.



This screenshot shows how MySQL Workbench lists my database tables on the left and how I can work on SQL queries over several tabs. This particular SQL query lists the number of portions provided each day in July 2023.

Outer frame

The graphical design is based on the wireframe sketches that I created during the meeting with [REDACTED]. The layout is intuitive with a menu down the left for primary navigation. A search box along the top can find people and other records. Icons in the top-right corner are for reports, settings and user account management. Every page will use this and the content for the page will be in the large central division.



```
function html_banner()
{
    html_add_logo();
    echo '<div class="banner">';
    html_add_search_box();
    html_add_icon('message-board.php', 'icon-message-board');
    html_add_icon('settings.php', 'icon-settings');
    html_add_icon('account.php', 'icon-account');
    echo '</div>';
}

function html_add_logo()
{
    echo '<div class="logo"></div>';
}

function html_add_search_box()
{
    echo '<form class="search" method="GET" action="search.php">
        <input type="text" name="find" autocomplete="off" placeholder="Search">
    </form>';
}

function html_add_icon($url, $icon_name)
{
    echo '<a href="' . $url . '" class="icon ' . $icon_name . '"></a>';
}

//HTML for the side bar menu, this uses the function below.
function html_menu()
{
    echo '<nav>';
    html_add_menu('home.php', 'Departments');
    html_add_menu('people.php', 'People');
    //html_add_menu('departments.php', 'Departments');
    html_add_menu('activities.php', 'Activities');
    html_add_menu('reporting.php', 'Reporting');
    html_add_menu('contact.php', 'Contact');
    echo '</nav>';
}

function html_add_menu($url, $menu_caption)
{
    echo '<a href="' . $url . '">' . $menu_caption . '</a>';
}
}
```

To the left is the code for the outer frame. Each function represents a part of the outer frame. All of these functions are then bundled together in another function called "start_html". To start any page all I need to do is open up a new empty file and call the function which will display the outer frame. By making use of include files and functions I can minimise all of the code for every other file.

People page

Clicking People in the left-hand menu shows the list of people that are eligible for food parcels.

Search

Departments

People

Activities

Reporting

Contact

People

Add Person

Name	Gender	Phone	Email
Bat man	Male		
	Male		
	Male		
Bilbo Baggins	Male	123456789	bilbobaggins@gmail.com
Black Widow	Female		
Bob Smith	Male		
Bugs Bunny	Male		
Cat Woman	Female		
Darth Vader	Male		
Forrest Gump	Male		
Freddy Krueger	Male		
God zilla	Male		
Han Solo	Male		

```
//get a list of people
$sql = "select
    person.PersonID,
    concat(person.FirstName, ' ', person.LastName) as Name,
    list_item.Name as Gender,
    person.Phone,
    person.Email
from person
left join list_item on list_item.Code = 'GEND' and list_item.ID = person.GenderID
order by concat(person.FirstName, ' ', person.LastName);";
$data = mysqli_query($conn, $sql);

//Show the results in a table
echo "<table class='listing'>
<thead>
<tr>
<td>Name</td>
<td>Gender</td>
<td>Phone</td>
<td>Email</td>
</tr>
</thead>
<tbody>";

while ($row = mysqli_fetch_assoc($data))
{
    //loop over each row from the database and create an HTML table row
    $id = $row["PersonID"];
    $name = $row["Name"];
    $gender = $row["Gender"];
    $phone = $row["Phone"];
    $email = $row["Email"];
    echo "<tr>
        <td><a href='person-view.php?person=' . $id . "'>$name</a></td>
        <td>$gender</td>
        <td>$phone</td>
        <td>$email</td>
        </tr>";
}
echo "</tbody>
</table>";
```

The code behind this is to the left. It starts by writing a SQL statement to return information from the database. It then creates a table for the data to be imputed, and lastly there is a while loop to loop over the data and place it in the correct table cells.

Viewing and editing a person

An operator can record an individual's information, like name, date of birth, ethnicity, address and dietary preferences.

Edit Person

MAIN DETAILS

Title:

First Name:

Last Name:

Date Of Birth:

Gender:

Ethnicity:

IDENTIFICATION

Drivers Licence:

NHI:

WINZ:

ADDRESS

Street:

Suburb:

City:

Post Code:

OTHER

Allergies:

Medical Conditions:

CONTACT DETAILS

Email:

Phone:

Other:

Comments:

EMERGENCY CONTACT

Name:

Phone:

Below is the code for the “Main Details” section. Include files are used again so all I have to do is call that function. There are different function names because there are different fields. Some are plain text boxes, drop down lists, dates and addresses. There are many options in the brackets that define certain aspects such as the width of the box, the description and the link to the database. For example on the third row the function is `form_edit_row`.

```
form_show_heading('Main Details');
form_drop_list_row($conn, 'Title:', 'TitleID', $person['Title']['ID'], 'TITL', EXTRA_NARROW_COLUMN);
form_edit_row('First Name:', 'FirstName', $person['FirstName'], EXTRA_WIDE_COLUMN, 45);
form_edit_row('Last Name:', 'LastName', $person['LastName'], EXTRA_WIDE_COLUMN, 45);
form_date_row('Date Of Birth:', 'DateOfBirth', $person['DateOfBirth'], EXTRA_NARROW_COLUMN);
form_drop_list_row($conn, 'Gender:', 'GenderID', $person['Gender']['ID'], 'GEND', EXTRA_NARROW_COLUMN);
form_drop_list_row($conn, 'Ethnicity:', 'EthnicityID', $person['Ethnicity']['ID'], 'ETHN', EXTRA_NARROW_COLUMN);
```

Below is the code for “`form_edit_row`” where we can see all of the options being put to use such as the caption and database_field.

```
function form_edit_row($caption, $database_field, $content, $width, $max_characters_allowed)
{
    echo "<tr class='field'>
        <td>$caption</td>
        <td><input type='text' name='$database_field' style='width:$width%' value='$content'
            maxlength='$max_characters_allowed' autocomplete='nope'></td>
    </tr>";
}
```

The data still needs to be received which is what this function does. By using the connection to the database and the person ID.

```
//Gets all information on a person and stores it in the variable '$person'
function get_person_info($conn, $person_id)
{
    //Selects all person info
    $sql = "select
        PersonID,
        TitleID,
        FirstName,
        LastName,
        GenderID,
        EthnicityID,
        DateOfBirth,
        Email,
        Phone,
        PhoneOther,
        Street,
        Suburb,
        City,
        PostCode,
        EmergencyContactName,
        EmergencyContactPhone,
        NHI,
        MedicalConditions,
        Allergies,
        DriversLicence,
        WINZ,
        Comments
    from Person
    where PersonID = '$person_id'";
    $data = mysqli_query($conn, $sql);
    $row = mysqli_fetch_assoc($data);
    if ($row == null)
    {
        return null;
    }
}
```

Add meal page


When viewing a person, an Add button lets the operator log a food parcel. The date of the parcel and the number of meals are entered. The code for this is the exact same concept as the edit and view person pages so it won't be repeated.

Code backups

I am backing up my code every time I do work on it to both my server and my home drive, and keeping snapshots from each day. If anything goes wrong, I can go back to an older version and check what caused the problem. If my hard drive fails, then the code is available in two other locations.

2022-07-17.database backup	20/07/2022 1:54 pm	File folder
2022-07-20.aws backup	26/07/2022 8:41 pm	File folder
2022-07-26.aws backup	26/07/2022 8:31 pm	File folder
2022-07-26.database backup	26/07/2022 8:32 pm	File folder
2022-07-27.aws backup	27/07/2022 7:23 pm	File folder
2022-07-27.database backup	27/07/2022 7:24 pm	File folder
2022-07-28.aws backup	28/07/2022 5:43 pm	File folder

This ensures my code is safe from accidental deletion or mistakes that might go unnoticed. However it is not as powerful as a version control system. I plan to switch to Github or another version control system in the near future.

I needed to organise my files in a logical order and make sure that all the files were named appropriately. This is because it would be more user-friendly and easier to navigate through. I have  as a folder containing all of the php files that are pages on the website and an "include" folder and a "resources" folder. **The Include Folder** contains all of the php files that are "includes" which are all functions. The resources folder contains my stylesheet document and a subfolder for images. By setting out the files like this with subfolders it's easy to keep track of all the files as they are all sorted and easy to find. By being organised I am better protected from losing my code either from messy files or not backing up. Obviously if I lose any code it's a major setback and I won't deliver my project on time to my stakeholder.

Testing phase 1

Description	Expected outcome	Actual outcome	Action from this	Result
Testing include files	That the "html_begin" function located in the "inc-html" file would run in other files if they included the "inc-html" file. This shows the search box, side menu, logo and icons. By making the parameter "Home" the title of the page will be Home	Everything went to plan	No action needed	Any file can have all the search box, side menu, logo and icons simply by calling the "html_begin" function. By making the parameter "Home" the title of the page will be Home

Screenshot 1: The function in inc-html.php

Screenshot 2: The function being called in home.php

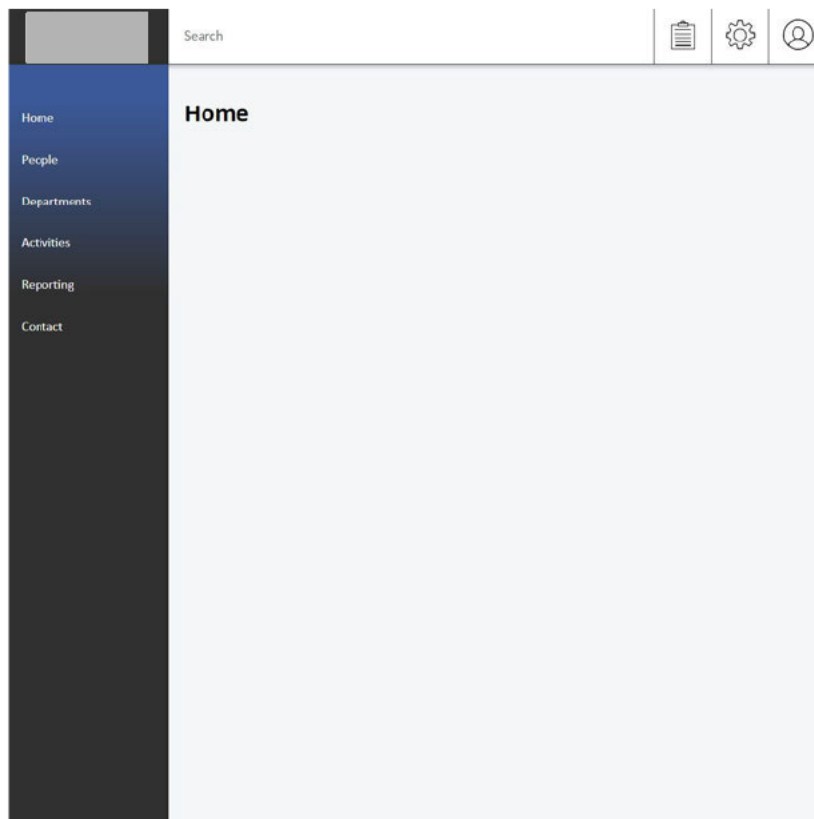
Screenshot 3: How it looks

```

56 function html_begin($page_title)
57 {
58     //runs html start
59     echo '<!DOCTYPE html>'
60     <html lang="en">
61     <head>
62     <title>' . $page_title . '</title>
63     <link rel="stylesheet" type="text/css" href="res/stylesheet.css">
64     <script src="res/javascript.js"></script>
65     </head>
66     <body>';
67     html_banner();
68     html_menu();
69     echo '<main>';
70     echo '<h1>' . $page_title . '</h1>';
71 }

```

```
6 html_begin('Home');
```



Testing connection to the database	Check that the four variables controlling the database connection work as expected (\$servername, \$username, \$password,	Connection to MySQL database worked	No action needed.	PHP code can connect to the database and therefore extract and update information
------------------------------------	---	-------------------------------------	-------------------	---

	\$dbname)			about people, list items and meals.
Blank name should be rejected	When adding a person and leaving the name blank, a message should prevent the record from being saved.	The record is saved with a blank name.	A check is needed for invalid first and last names.	Saving a person with a blank name breaks the list on the People page. The link is not visible when the name is blank.
Test that text fields are edited and saved properly.	When editing a text field, like Phone, make sure that the value typed is saved in the database.	After saving a person, viewing them should show the phone number that was typed in.	It works as expected.	Text fields are created by the function "form_edit_row" which is used for every field.
Test that drop lists show the right items	When editing a person, the Gender and Ethnicity drop lists should come from the database.	The items should match records in the list_item table	They match correctly, but when editing a person they didn't default to the correct value. I changed the code so that they do by adding a \$selected_id parameter	Drop lists are based on database rows, making it easy for users to change them.
Check that each field is stored in the correct database column	When saving a person, make sure that the name is stored in the name column, and that ethnicity is stored in the ethnicity column, etc.	All fields worked except ethnicity due to a copy/paste error	The code was writing ethnicity to the gender column. I fixed the PHP code and tested it again.	Saving a person puts all the fields in the right place.
Add meal	When clicking add meal, make sure that the date and recipient	Saves correctly	No action needed	

	are saved correctly in the database.			
Date formatting checks	When editing a meal recipient, the date it was picked up must be entered in a proper date format.	Entering an invalid date like "hello" causes the field to be blank.	The field needs to be checked when clicking Save to see that it is a proper date.	Meal pick ups need a date for correct reporting.
SQL instructions entered by the user in fields should not break the system	When editing a person's name, typing SQL command hacks should be rejected.	Adding a name like "Bob';delete from Person" causes the SQL to be run, breaking the system	Checks are needed that names and other values entered by the user are safe.	At the moment, it is possible to break the database. Will need to look at this in the future but it looks like there's no easy fix now and more important things to do at the moment. The only people that are able to access the system are DCST staff and they would never input a detailed SQL command to break the system so this feature is not that important but still needs to be looked at in the future.
Emojis and foreign language characters	Typing emojis and characters in Japanese, Arabic, etc should be accepted	They are replaced with question marks (?)	Database fields need to be changed from	When fixed, saving emojis and other characters

			VARCHAR to NVARCHAR (which has unicode support)	will work properly.
--	--	--	---	---------------------

Add Meal

Edit

MAIN DETAILS		IDENTIFICATION	
Title:	Mr	Drivers	12345
Name:		Licence:	
		NHI:	123

Teacher login.	To test the system, a login is needed for . Create a user called with a password and check that the login works.	Logging in to should accept and password	Logging in with these credentials works.	This allows markers to access the system without making the system public.
----------------	--	--	--	--

Sign in to access this site
 Authorization required by
 Username
 Password

Sign in

Cancel

Feedback of work so far

At this stage I decided to share my work with I and get feedback/feedforward to ensure I am on the right track. This was 's reply:

First of all, thank you again for choosing to use for your project case study. We are grateful for your interest in what we do, and for your commitment to visit us, examine our current database, and learn the changes and upgrades we would require for optimal functionality.

Obviously many of the system changes we would like were beyond the scope and timeline of your school project, and you informed us of that. However, the work you have completed so far has given us a taste of what is possible, and has helped us to better visualise our bigger project picture. If you have set up this database as if it is just used for our Foodbank, when in the future we can see this able to be used across all our programmes and departments.

My colleague, [REDACTED] (who works in our Foodbank and uses the database the most), and I looked through your database together and have the following feedback: You have created a database that is visually much more appealing than our current one. We like the layout of the main headings down the left hand side and the large search bar at the top.

Adding a person is easy to do. I like the layout, and how you have grouped different things together under headings. You have included the drop down boxes we asked for (Title, gender, ethnicity). Perhaps the date of birth could give more of an idea what format the information is required in to stay consistent? I see that once you add a new person it then translates their date of birth automatically into an age - which is really helpful.

We like having the 'add meal' button. It's great that it populates the date automatically, but also can be overwritten if we needed to put in an earlier date.

Other next steps would be to include a search engine for the street address, so that when you start typing it brings up the address automatically, and then have it automatically confirm if they are in our eligible area or not. And then to be able to continue this model on for all our other programmes and departments, as well as giving reporting capabilities.

I know we have talked about the reporting before and that is yet to come. We are excited to see what you will create.

So, in summary, [REDACTED] and I agree you have understood what we were looking for in a new database, and made excellent progress towards that.

We are grateful for the time and energy you have put into this project, and look forward to continuing working with you if you have the time and inclination.

I would be very happy to discuss any of this further with your teacher if they require it.

Thanks again [REDACTED]

Kind regards,
[REDACTED]

Deconstructing feedback

Feedback	Actions
Date formatting	When editing forms that include “Date” fields like “Date of Birth”, there is no hint as to how the date should be formatted. In another conversation [REDACTED] mentioned having a pop-up window when the date field is clicked that would contain a calendar to select a date. This would fix formatting issues and make everything easier.
Reporting	The reporting side of this project is huge and I am yet to start on it however I have pointers from the original brief of what to do
Multiple Departments	[REDACTED] wants support for their other departments namely Friendship Link and Evergreen club, programmes for intellectually disabled adults and elderly people. This would involve different features like a roll call and attendance tracker. This is a lower priority than the foodbank functions as it's more important to get that working and move onto the next thing.
Automatic addresses	When typing in an address, [REDACTED] would like it to automatically fill in all fields with the correct address, and find out if the person lives in an eligible area. I'll find out how websites do this, and integrate it into the system.

Phase 2 development

Calendar system

One of the bits of feedback I got on phase one's prototype was that when entering a date field in a person information page there was no indication as to what date format should be used. Further than that [REDACTED] asked if it was possible to have a pop-up calendar instead manually typing dates in. This would be a lot easier for them as typing out dates for every new person that joins [REDACTED] is needlessly time-consuming. I asked my technologist in my interview and he said to look into “Materialize CSS”

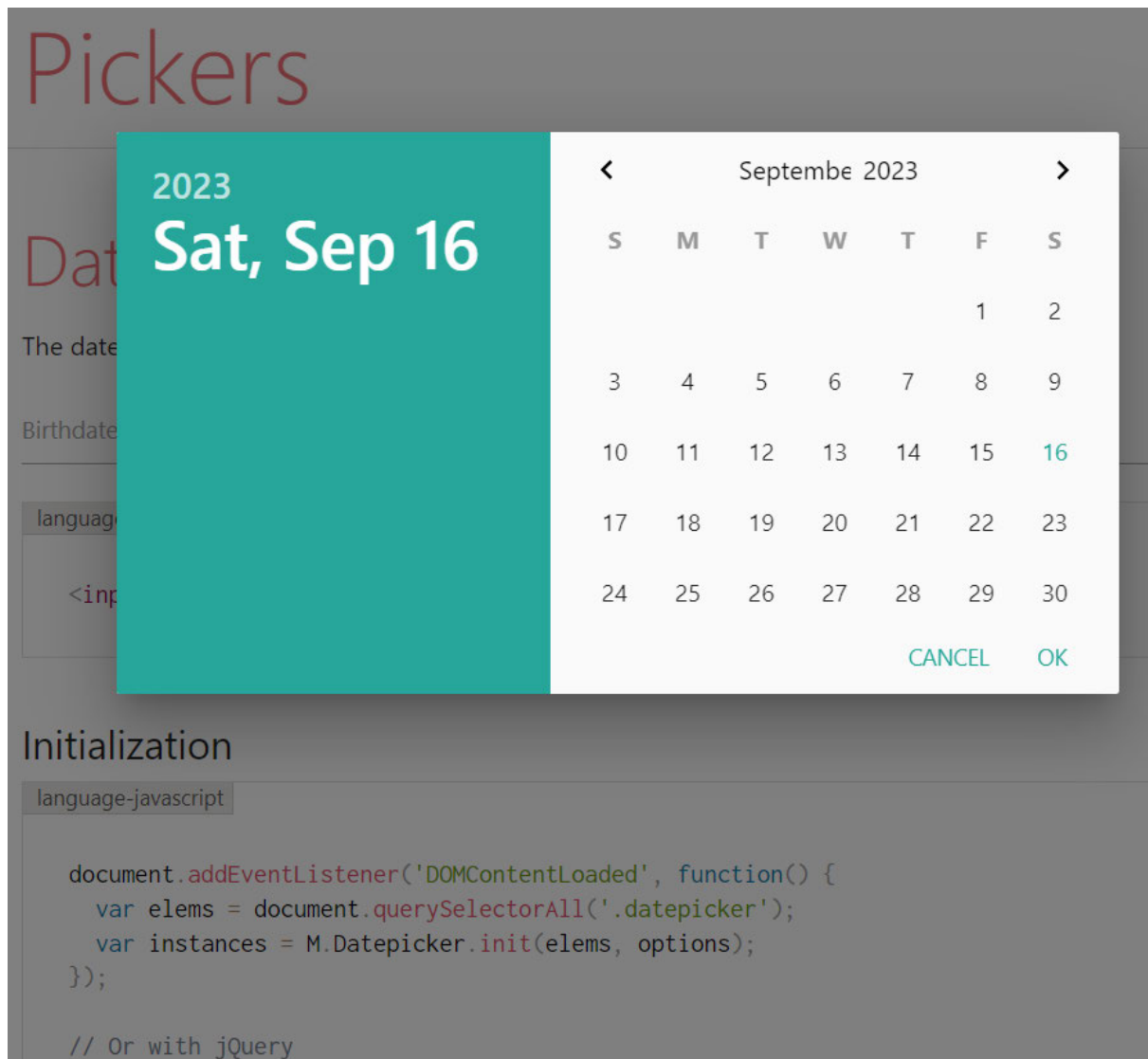
Materialize CSS

This is a web development framework that can be incorporated into any website. The benefits are that it has useful components like calendars, tabs, popup windows, and other user interface widgets.

The calendar widget is the most relevant, and can convert a normal text input box to an interactive calendar in a popup window.

Here is an example from the MaterializeCSS website at

<https://materializecss.com/pickers.html>



This looked like a perfect solution to [redacted]'s request for a date picker. I researched how to include MaterializeCSS into my project, which is with this code:

```
<link rel="stylesheet" type="text/css" href="res/materialize.min.css">
<script src="res/materialize.min.js"></script>
```

MaterializeCSS needs two files, one javascript file and one CSS file. I downloaded both of these, copied them into my project, and included them in my header file.

It immediately created a major problem, in that the entire styling of my project was overridden. Colours were gone, font sizes were wrong, positioning and spacing was wrong, and some elements like dropdown lists disappeared altogether. Borders were added where I didn't want them. The problems were due to the CSS file from MaterializeCSS. It can't be removed, as the calendar wouldn't look right, and can't be easily edited. My solution was to add an extra CSS file to override the parts of MaterializeCSS that I don't want. I called the file "materialize-overrides.css". I used the browser inspector to reverse engineer the CSS rules and work out what CSS overrides I needed to add.

The result was a 100-line CSS file, mostly dealing with colours, borders, and padding. Here are the first 50 lines of the file:

```
/*materialize-overrides.css*/
/*This file contains all of the styling that overrides the materialize.css default css*/
:root {
  --accent-color: #3b5998;
  --accent-color-text: white;
  /*Delta Orange #e89a00*/
}

/*General styling preferences to override materialise css*/
button:focus {
  background-color: var(--accent-color);
}

tr {
  border-bottom: none;
}

td {
  line-height: normal;
}

/*Remove visual effects from the main search box*/
form.search > input[type=text] {
  border-radius: 0!important;
}

form.search input[type=text]:focus {
  box-shadow: none!important;
  border-bottom-color: black!important;
}

/*Undo materialise changes to form fields*/
table.form tr.field > td:last-child > select {
  display: initial;
  height: auto;
}

table.form tr.field > td:last-child > input[type="text"] {
  height: inherit;
}

table.form tr.field > td:last-child > input[type="text"]:focus {
  box-shadow: 0 1px 0 0 var(--accent-color);
}

/*Change the color of drop down selects*/
.dropdown-content li span {
  color: black;
}
```


I included the CSS file immediately after the main MaterializeCSS file, so that its CSS rules take priority:

```
<link rel="stylesheet" type="text/css" href="res/materialize.min.css">
<link rel="stylesheet" type="text/css" href="res/materialize-overrides.css">
<link rel="stylesheet" type="text/css" href="res/stylesheet.css">
```

The lowest priority stylesheet rules is the official MaterializeCSS file. The next priority is my overrides to undo the undesired rules. The third file has top priority, which is my main CSS file.

Here are before and after screenshots, showing how MaterializeCSS added undesirable borders and spacing:

The screenshot shows a web application interface. On the left is a dark blue sidebar with navigation links: Departments, People, Activities, Reporting, and Contact. The main content area is titled 'Edit Person'. It contains several sections: 'MAIN DETAILS' with fields for Title, First Name (Bilbo), Last Name (Baggins), Date Of Birth, Gender, and Ethnicity; 'IDENTIFICATION' with fields for Drivers, Licence, NHI, and WINZ; 'ADDRESS' with fields for Find (Search) and Street; and 'OTHER' with fields for Allergies, Medical Conditions, and Comments. The form has a search bar at the top and a settings icon at the bottom right.

With the overrides in place, it looks the way it did before using MaterializeCSS:

Search

Edit Person

MAIN DETAILS

Title:

First Name:

Last Name:

Date Of Birth:

Gender:

Ethnicity:

IDENTIFICATION

Drivers Licence:

NHI:

WINZ:

OTHER

Allergies:

Medical Conditions:

ADDRESS

Find:

Street:

Suburb:

City:

Comments:

It was a major inconvenience getting MaterializeCSS integrated into my project. The overrides took a long time, and made me question the advice from my technologist. I think there are probably more suitable calendar components available that would be better. The problem with MaterializeCSS is that it's not just a calendar. It's an entire design template that needs to be followed right from the start. Looking back, it would have been a lot better to use a simple calendar component from the web. Anyway, I managed to get MaterializeCSS working well, so decided to stick with it as it has many other components like tabs, buttons with icons, searchable drop lists. These are things I would like to use in a future version of the project.

Converting the Date of Birth field to use the MaterializeCSS calendar required a few lines of HTML and javascript. For consistency I wrote a function "form_date_row" so that the original text box can be easily changed to a calendar by using a different function:

```
//simple text box
form_edit_row('Date Of Birth:', 'DateOfBirth', '', EXTRA_NARROW_COLUMN, 20);
//awesome popup calendar
form_date_row('Date Of Birth:', 'DateOfBirth', '', EXTRA_NARROW_COLUMN);
```

The function "form_date_row" shows a field on screen that when clicked gives a popup calendar. The code puts an input box on the screen, and waits for the page to fully load. Once loaded, it converts the input box to a calendar using the M.Datepicker.init(), which is described in the MaterializeCSS documentation. The function takes two parameters, one is the input box and the other is options that control its behaviour. The options I specified are the NZ date format where the day is

before the month, and other options like a close button and that the calendar should auto close when clicking away.

```
function form_date_row($caption, $database_field, $content, $width) {
    echo "<tr class='field'>
        <td>$caption</td>
        <td>
            <input type='text' name='$database_field' value='$content' style='width:$width%'>
            <script>
                document.addEventListener('DOMContentLoaded', function()
                {
                    var cssRule = 'input[name=$database_field]';
                    var input = document.querySelector(cssRule);

                    var options = {
                        format: 'd mmm yyyy', // Date format
                        autoClose: true, // Close the picker when a date is selected
                        showClearBtn: true, // Show a 'Clear' button next to the picker
                        firstDay: 1,
                        yearRange: [1923, (new Date()).getFullYear()]
                    };
                    M.Datepicker.init(input, options);
                });
            </script>
        </td>
    </tr>";
};
```

The result is that the input box still looks the same, but when clicked a large easy to use calendar appears:

The screenshot shows a web application interface with a sidebar on the left containing links for Departments, People, Activities, Reporting, and Contact. The main content area is titled 'Edit Person' and contains two columns of form fields. The left column, labeled 'MAIN DETAILS', includes fields for Title, First Name (Bilbo), Last Name (Baggins), Date Of Birth, Gender (Male), and Ethnicity. The right column, labeled 'IDENTIFICATION', includes fields for Drivers Licence, NHI, WINZ, and OTHER. The 'Date Of Birth' field is highlighted with an orange rectangle, and a large, user-friendly calendar date picker is displayed over it, showing the month of January and the year 2023. The calendar has a close button and an auto-close feature.

The top screenshot shows a web application interface with a sidebar menu containing 'Departments', 'People', 'Activities', 'Reporting', and 'Contact'. A search bar is at the top. A date picker is open, showing a calendar for September 2023. The date 'Sat, Sep 16' is selected. The bottom screenshot shows the 'Edit Person' form. The 'Date Of Birth' field is highlighted with an orange box, showing '16 Jul'. The form includes sections for 'MAIN DETAILS' (Title, First Name, Last Name, Date Of Birth, Gender, Ethnicity) and 'IDENTIFICATION' (Drivers Licence, NHI, WINZ, OTHER, Allergies, Medical Conditions).

Clicking the year shows a droplist of dates, as does the month. This makes it easy to pick a date with the mouse. Once a date is selected, it appears in the text box. The value is then saved in the database. Details about database reading and writing are below.

Conclusion

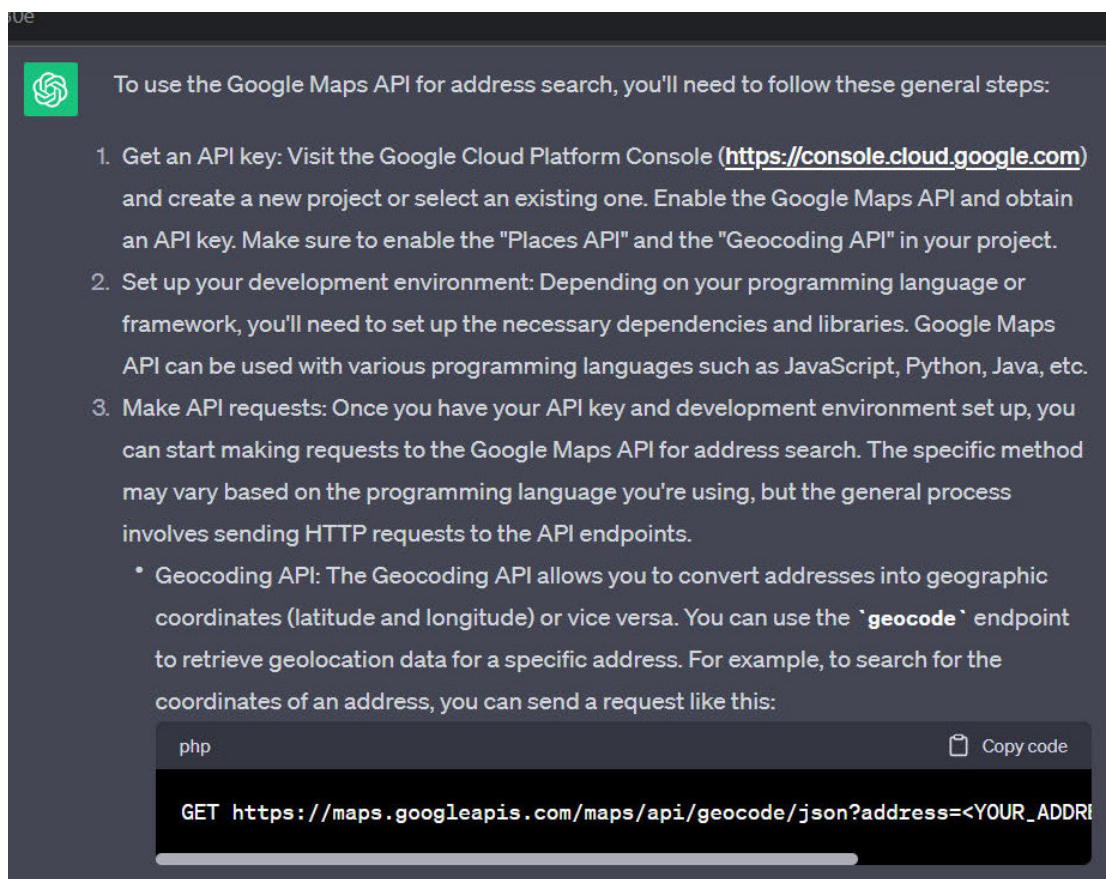
The implementation of MaterializeCSS has proven to be a valuable solution to address [redacted]'s request for a date picker in the person information page. This addition not only fulfils their immediate need for a user-friendly date entry system but also sets the stage for future enhancements in the project. While the integration of MaterializeCSS came with its fair share of challenges, it ultimately aligns with [redacted]'s stakeholder requirements. MaterializeCSS opens doors for future enhancements beyond the date picker. The framework offers a range of components, including tabs, buttons with icons, and searchable dropdown lists. These components can be used to improve the overall user interface and user experience in future versions of the project.

Address autofill

Another feature asked for when giving feedback was address auto-filling. This feature would be used when staff are adding an address to a person's information. Manually filling address information is tedious and prone to human errors and not very user friendly, so an automated option would be a big improvement. They asked if there was a way so that as you type an address there is an auto-complete which when clicked auto-fills the address fields. After consulting with my technologist, he recommended using the Google Maps API.

Challenges and Considerations

Using Google APIs is not part of my skill set. When researching I only found very broad information because the Google Maps API is used for so many things there was nothing about implementing it as a search engine to find addresses. I tried asking ChatGPT the steps I need to take to implement the API in my situation. This is what it replied with:



The screenshot shows a ChatGPT interface with a dark theme. At the top left is the OpenAI logo. The text of the response is as follows:

To use the Google Maps API for address search, you'll need to follow these general steps:

1. Get an API key: Visit the Google Cloud Platform Console (<https://console.cloud.google.com>) and create a new project or select an existing one. Enable the Google Maps API and obtain an API key. Make sure to enable the "Places API" and the "Geocoding API" in your project.
2. Set up your development environment: Depending on your programming language or framework, you'll need to set up the necessary dependencies and libraries. Google Maps API can be used with various programming languages such as JavaScript, Python, Java, etc.
3. Make API requests: Once you have your API key and development environment set up, you can start making requests to the Google Maps API for address search. The specific method may vary based on the programming language you're using, but the general process involves sending HTTP requests to the API endpoints.
 - Geocoding API: The Geocoding API allows you to convert addresses into geographic coordinates (latitude and longitude) or vice versa. You can use the `'geocode'` endpoint to retrieve geolocation data for a specific address. For example, to search for the coordinates of an address, you can send a request like this:

php Copy code

```
GET https://maps.googleapis.com/maps/api/geocode/json?address=<YOUR_ADDRESS>
```

Replace `**<YOUR_ADDRESS>**` with the address you want to search for, and `**<YOUR_API_KEY>**` with your actual API key. The response will contain the latitude and longitude of the address.

- Places API: The Places API allows you to search for places based on various criteria, including address. You can use the `**textsearch**` or `**nearbysearch**` endpoints to search for places based on a query string or proximity to a location. For example, to search for places near a specific address, you can send a request like this:

php

Copy code

```
GET https://maps.googleapis.com/maps/api/place/nearbysearch/json?location
```

Replace `**<LATITUDE>**` and `**<LONGITUDE>**` with the coordinates of the address you want to search around, `**<RADIUS>**` with the search radius in meters, and `**<YOUR_API_KEY>**` with your API key. The response will contain a list of places matching your search criteria.


4. Process the API response: Once you receive the response from the API, you can parse the data and extract the relevant information for your application. The response will typically be in JSON format, which you can handle using the appropriate JSON parsing library for your programming language.

The results from ChatGPT were out of date, but it helped point me in the right direction, which was to the official Google documentation of the Maps Javascript Place Autocomplete API.

<https://developers.google.com/maps/documentation/javascript/place-autocomplete>

The Google webpage has some sample code which I used as a starting point.

```
const center = { lat: 50.064192, lng: -130.605469 };
// Create a bounding box with sides ~10km away from the center point
const defaultBounds = {
  north: center.lat + 0.1,
  south: center.lat - 0.1,
  east: center.lng + 0.1,
  west: center.lng - 0.1,
};
const input = document.getElementById("pac-input");
const options = {
  bounds: defaultBounds,
  componentRestrictions: { country: "us" },
  fields: ["address_components", "geometry", "icon", "name"],
  strictBounds: false,
};
const autocomplete = new google.maps.places.Autocomplete(input, options);
```


The sample code was not perfect for my case. I changed it to search  addresses only, and return only the address fields that I need (ie. street, suburb, city and postcode). The sample code also doesn't split the address into separate fields, so I wrote code to do this.

The first part of my code includes the Google Maps Javascript API into my code. In order to use the Google Maps API, an API key is needed. To get my key, I had to go through a process on the Google website. It was fairly straightforward and gave me a key to use, which is about 40 characters long. I've blurred it in the screenshot, as Google requires the API keys they issue to be kept confidential:

```
//connects to js file and google maps api
$api_key = 'AIzaSyDxQwm68uMmUwAsIvsaU0BdFyB677hy4B1t6a';
echo '<script defer src="https://maps.googleapis.com/maps/api/js?key=' . $api_key
    . '&callback=initAutocomplete&libraries=places&v=weekly"></script>';
```

The second part was to create a javascript file to manage the API, which I called "address-finder.js". The first function sets up an input box to act as the address search field.

```
function addAutocompleteToAddressField() {
    //convert the address search box into a Google Places API search auto completer
    var input = document.querySelector("input[name=PlaceFinder]");
    input.setAttribute('placeholder', 'Search');
    //create a geographic boundary for Canterbury
    var southWestPoint = new google.maps.LatLng(-43.785,171.947); //Rakaia
    var northEastPoint = new google.maps.LatLng(-43.143,173.244); //Gore
    var searchBoundary = new google.maps.LatLngBounds(southWestPoint, northEastPoint);
    //work out what options to tell google that we want
    var options = {
        componentRestrictions: { country: 'nz' },
        fields: ["address_components", "formatted_address", "geometry"],
        types: ["address"],
        strictbounds: true,
        bounds: searchBoundary
    };
    //turn the input box into an autocomplete box
    var autocomplete = new google.maps.places.Autocomplete(input, options);
    //listen for when a person selects an address
    autocomplete.addListener("place_changed", () => { userDidChoosePlace(autocomplete.getPlace()); });
}
```

This code has similarities to the Google sample code, but these additions:

- Only returns addresses, as opposed to places of interest
- Only matches addresses in central Canterbury (south of Gore, north of Rakaia)
- Has an event handler that gets called when the user selects an address

The event handler takes the selected address, extracts the relevant address components (street, suburb, city, postcode) and puts them in the corresponding address fields.

```
function userDidChoosePlace(place) {
  //called when the user selects an address from the drop down list of addresses returned by the Google Places API
  if (place.geometry && place.geometry.location) {
    //console.log(place.formatted_address);
    //console.log(place.address_components);
    var streetNumber = findAddressComponent(place.address_components, "street_number");
    var street = findAddressComponent(place.address_components, "route");
    var suburb = findAddressComponent(place.address_components, "sublocality");
    var area = findAddressComponent(place.address_components, "locality");
    var city = findAddressComponent(place.address_components, "administrative_area_level_1");
    var postCode = findAddressComponent(place.address_components, "postal_code");
    document.querySelector("input[name=Street]").value = joinStringsIfDifferent(streetNumber, street);
    document.querySelector("input[name=Suburb]").value = joinStringsIfDifferent(suburb, area);
    document.querySelector("input[name=City]").value = city;
    document.querySelector("input[name=PostCode]").value = postCode;
    document.querySelector("input[name=PlaceFinder]").value = "";
  }
}
```

Extracting the relevant information for individual address fields was problematic. The data returned from Google is an array of “type” = “value”, where type can be street_number, area, postal_code, etc. The problem is that there are multiple answers for some of these fields. I wrote a function “findAddressComponent” to simplify the process of extracting address components:

```
function findAddressComponent(data, key) {
  //given data returned by the Google Places API, look for a value with the given key
  for(var index = 0; index < data.length; index++) {
    var item = data[index];
    if (item.types.join(',').indexOf(key) >= 0) {
      return item.long_name;
    }
  }
  return "";
}
```

The function takes the data from Google and the type of data needed, like “postal_code”. It loops over all the data looking for the first value that matches the type and returns it as a string.

The function is used several times to get simple variables like street, suburb, postCode, etc.

Another problem I found is that parts of the address information returned by Google had duplicate information. For example, it would return an address with the suburb and area both as Bishopdale, and other addresses where the suburb and area are different. To avoid having duplicate information in the address, I wrote a function “joinStringsIfDifferent” to make sure that duplicate data is removed.

```
function joinStringsIfDifferent(string1, string2) {
  //given two strings, join them with a space if they are
  //different (if they are the same then just return one of them)
  if (string1 != string2) {
    return string1 + ' ' + string2;
  }
  return string1;
}
```


Once the javascript files were ready, I added a field to the edit person page, called “PlaceFinder” with this code. It uses the functions from phase one’s prototype to place form fields on the screen with single lines of code:

```
form_show_heading('Address');
form_edit_row('Find:', 'PlaceFinder', '', EXTRA_WIDE_COLUMN, 45);
form_edit_row('Street:', 'Street', '', EXTRA_WIDE_COLUMN, 45);
form_edit_row('Suburb:', 'Suburb', '', EXTRA_WIDE_COLUMN, 45);
form_edit_row('City:', 'City', '', MEDIUM_COLUMN, 20);
form_edit_row('Post Code:', 'PostCode', '', EXTRA_NARROW_COLUMN, 10);
```

This code follows the same pattern as other edit fields for a person, which are already saved in the database. The last parameter (ie. 45, 20, 10) determines the maximum number of characters the database field can store. They prevent the user entering longer information than the database can support.

No further changes were needed, as the address auto-complete feature relies mostly on the javascript code.

The result is that the user can start typing an address, and the top matches in the Canterbury region are shown. In this example, “123 F” is typed.

The screenshot shows a form titled "ADDRESS" with five input fields: "Find:", "Street:", "Suburb:", "City:", and "Post Code:". The "Find:" field contains the text "123 F". Below the "Find:" field, a dropdown menu is open, displaying five suggestions, each preceded by a location pin icon. The suggestions are: "123 Fe [redacted] New Zealand", "123 F [redacted] ew Zealand", "123 F [redacted] New Zealand", "123 F [redacted] New ...", and "123 F [redacted] New Zealand".

Selecting the first item on the list hides the list and fills in the address components. When the user clicks Save, this information is then saved in the “person” table. There are columns in the table called “Street”, “Suburb”, “City” and “PostCode” which match the fields on screen. More information about reading and writing to the database can be found in the Database Querying section, below.

The screenshot shows the same "ADDRESS" form, but the dropdown menu is now closed. The "Find:" field now contains the text "Search". The "Street:" field contains "123 [redacted]". The "Suburb:" field contains "[redacted]". The "City:" field contains "[redacted]". The "Post Code:" field contains "[redacted]".

Conclusion

The new address auto-fill feature will hopefully help [redacted] reduce the time and annoyance it takes to fill out client information and keep all of the data consistent by not allowing for errors in typing. My technologist advice and research using AI was instrumental in the development of this feature, proving to be very helpful, I might be able to use this again in later stages of the development to create a better end product at a faster rate for my stakeholder.

Database planning reflection

The process of planning a relational database through the visual representation of tables and their relationships proved to be a big help in the development phase. By planning out tables and their connections, I was able to create a structured framework that enhanced data organisation, making it easier to manage and comprehend. This approach not only improved the overall efficiency of the database but also allowed me to carry out the development process following a plan, saving time and effort. This means I can meet my stakeholder's needs faster and better.

The code has 21 queries on the database to support all of the features. For example showing the dashboard with drill down support, editing a person, adding meal portions, and searching the system. Below are three examples.

```
//Access database and return an array of departments that the person belongs to
function get_person_departments($conn, $person_id)
{
    $result2 = [];
    $sql = "select i.ID, i.Name
            from person_link_department d
            inner join list_item i on i.code = 'DEPT' and i.ID = d.departmentID
            where PersonID = $person_id
            order by position;";
    $data = mysqli_query($conn, $sql);
    while ($row = mysqli_fetch_assoc($data))
    {
        $id = $row["ID"];
        $name = $row["Name"];

        array_push($result2, [
            'DepartmentID' => $id,
            'DepartmentName' => $name
        ]);
    };
    return $result2;
}
```

This example lists the IDs and names of the departments that a person belongs to. The names of each department are stored in the list_item table, and the connection between people and their departments is stored in the person_link_department table. The query uses a join to link two tables. This allows for a many to many relationship between people and departments. It also means that new departments can be easily added by simply adding a new row to the list_item table. This works towards future proofing the system.

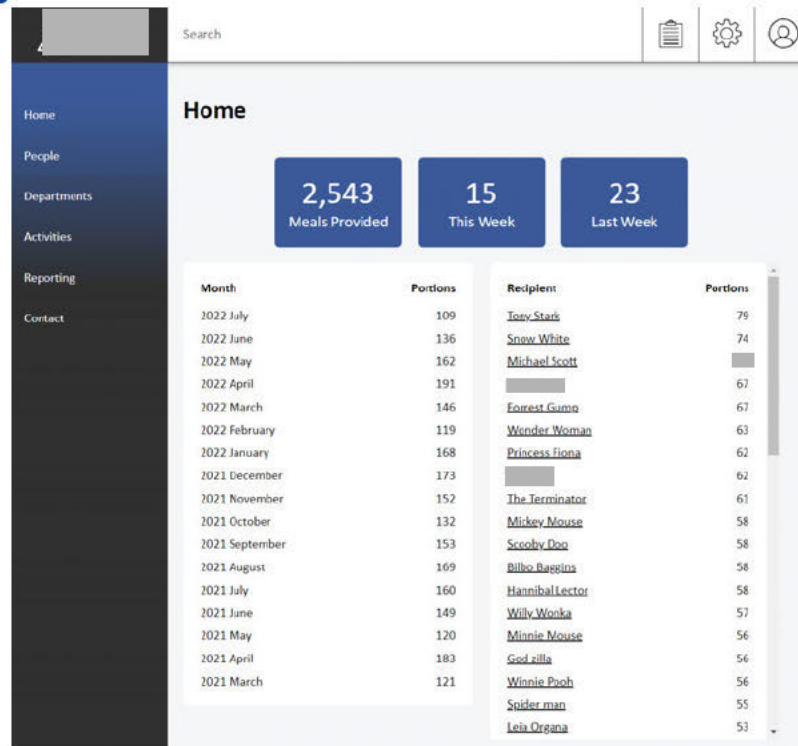
```
//Gets and displays recipients of meals on home page
function display_meal_recipients_in_date_range($conn, $from, $to)
{
    $from_string = date('Y-m-d', $from);
    $to_string = date('Y-m-d', $to);
    $sql = "select
        Person.PersonID,
        concat(Person.FirstName, ' ', Person.LastName) as Name,
        sum(meal_recipient.Portions) as Total
    from Person
    inner join meal_recipient on meal_recipient.PersonID = Person.PersonID
    where Date >= '$from_string' and date <= '$to_string'
    group by Person.PersonID, concat(Person.FirstName, ' ', Person.LastName)
    order by Total desc;";
    $data = mysqli_query($conn, $sql);
}
```

This example takes a date range as a parameter and returns all people that have received a meal in the time period and the number of meals. The query uses grouping to aggregate multiple rows to the level of single people.

```
//Rewrites a database field
function save_person_field($conn, $id, $database_field, $value)
{
    $sql = "update Person
        set $database_field = '$value'
        where PersonID = '$id'";
    mysqli_query($conn, $sql);
}
```

This function updates a single field for a single person. For example it sets the phone number to a value. This query uses an update statement to make changes to the database rows.

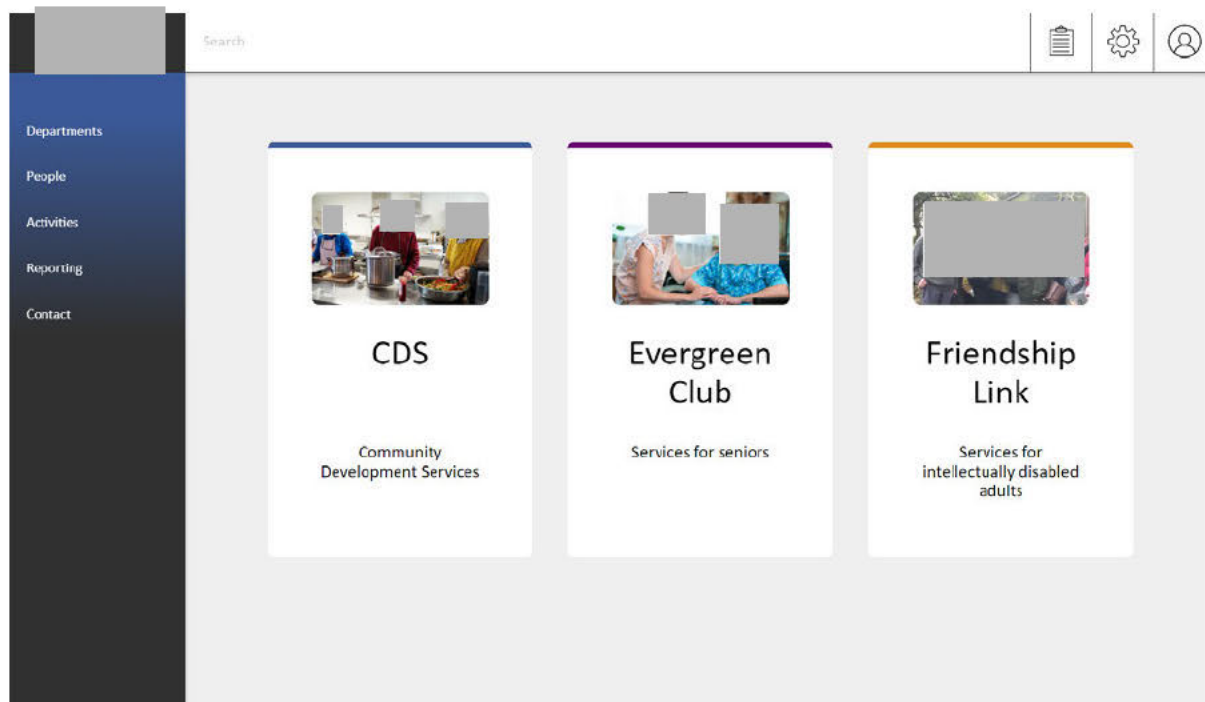
Home page



After logging in, Delta users see the system's home page. It lets users quickly navigate to the relevant part of the system:

- CDS (Community Development Services), the food bank
- Evergreen Club, support groups for seniors
- Friendship Link, support services for people with disabilities

The current state of the project has the CDS dashboard completed, with Evergreen Club and Friendship Link dashboards coming soon.



When designing the home page, I wanted an aesthetically pleasing design with cards for each department. The cards are colour coded:

- blue for the food bank dashboard,
- purple for the Evergreen Club, and
- orange for the Friendship Link

The completed CDS dashboard uses a blue theme, and purple and orange will be used for the other two.

The idea with colours is to make the system easy to navigate, because users will know at a glance what part of the system they are in due to the colour scheme.

To implement the cards, I created a function "create_dashboard_card". It takes 4 parameters:

- the target page to load when clicking the card
- the URL of the image to show
- the header text
- the footer text

Using a function means that there is no unnecessary duplication of code, as each card is made with one line of code.

```

echo("<div class='home-page-cards'>");
create_dashboard_card("cds.php",
    "res/img/cds.jpg",
    "CDS",
    "Community Development Services");
create_dashboard_card("evergreen.php",
    "res/img/evergreen.jpg",
    "Evergreen Club",
    "Services for seniors");
create_dashboard_card("friendshiplink.php",
    "res/img/friendship.jpg",
    "Friendship Link",
    "Services for intellectually disabled adults");
echo("</div>");

```

The function creates a small number HTML elements for each card:

```

//creates dashboard card for the home page with a picture title and footnote
function create_dashboard_card($url, $picture, $title, $subtext) {
    echo("<div class='dashboard-card'>
        <a href='$url'>
            <img src='$picture'>
            <h3>$title</h3>
            <h4>$subtext</h4>
        </a>
    </div>");
}

```

The cards use CSS and flexbox for a balanced layout. The row of three cards uses flexbox to get them centred and equally spaced. The inside of each card also uses flex so that the footer starts at the same height no matter how big the card titles are.


```
/* ----- HOME PAGE ----- */

/*flex box rules for card layouts*/
.home-page-cards {
  display: flex;
  justify-content: center;
  gap: 40px;
  padding-top: 40px;
}

/*styling for a single card*/
.dashboard-card {
  display: flex;
  flex-direction: column;
  background-color: white;
  align-items: center;
  width: 300px;
  padding: 50px 50px 50px 50px;
  border-top: 7px solid black;
  border-radius: 7px;
}

/*color for top border of each card*/
.dashboard-card:nth-child(1) {
  border-top-color: #3b5998;
}

.dashboard-card:nth-child(2) {
  border-top-color: #6A0572;
}

.dashboard-card:nth-child(3) {
  border-top-color: #dc8c1c;
}

/*the styling for the card image*/
.dashboard-card img {
  width: 200px;
  border-radius: 8px;
  display: block;
  margin: 0 auto;
}

/*header text*/
.dashboard-card h3 {
  font-size: 40px;
  text-align: center;
  min-height: 100px;
}

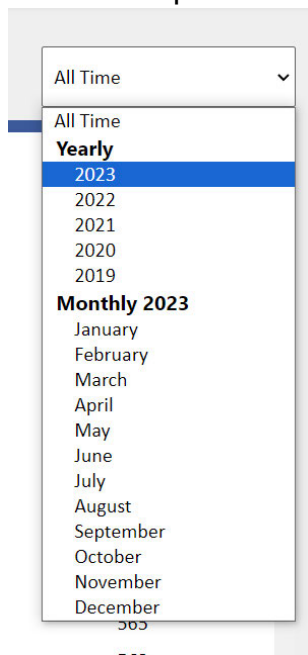
/*footer text*/
.dashboard-card h4 {
  font-size: 20px;
  text-align: center;
}
```

In summary, the combination of the PHP function, HTML elements and the flexbox CSS is a tidy and convenient way of implementing the card layout.

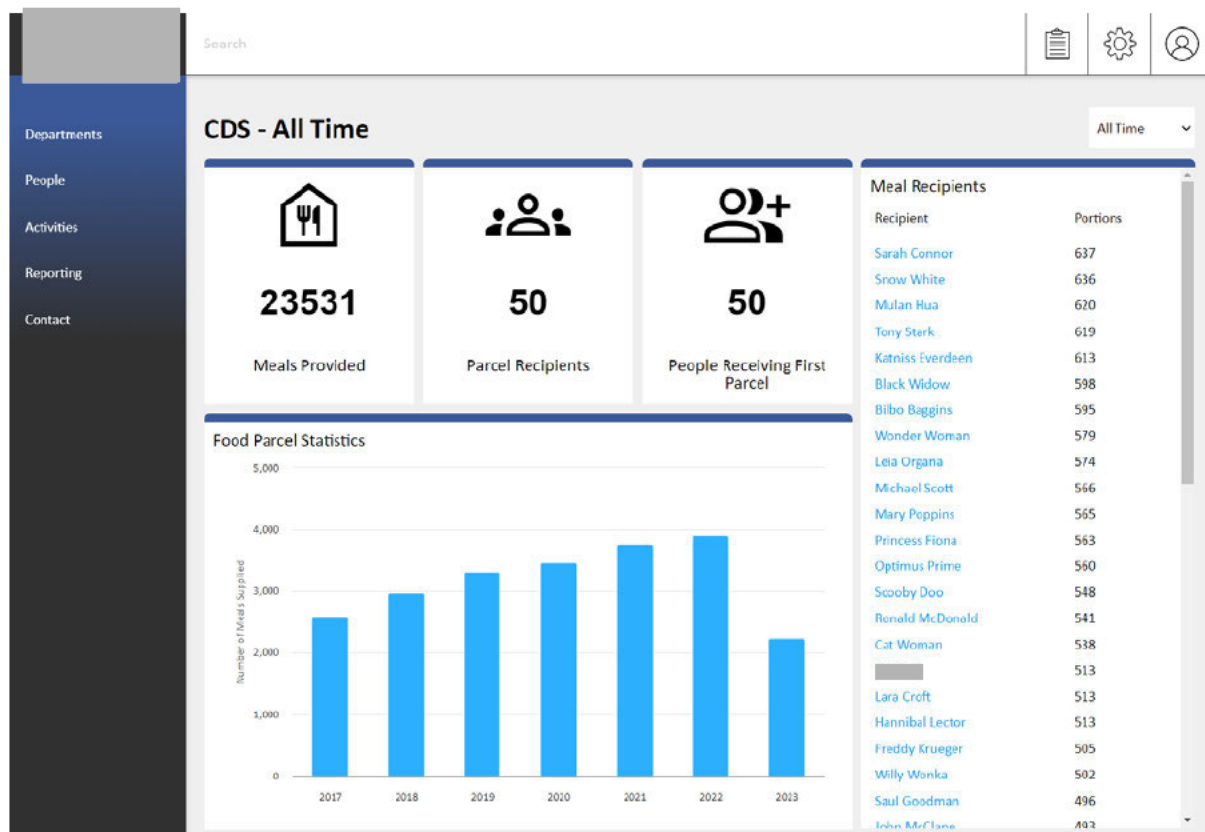
CDS dashboard

The CDS (Community Development Services) Food Bank Dashboard is the solution to one of [redacted]'s largest requirements, which is to provide statistics to its funders. The dashboard is the starting page for the food bank part of the system. It is a powerful reporting tool with a flexible time period selector. There are three levels of drill down data analysis available.

The funders require data around food parcels over time, and the time selector makes this possible. It is a drop down list that has both years and months to choose from as well as an option for all time.

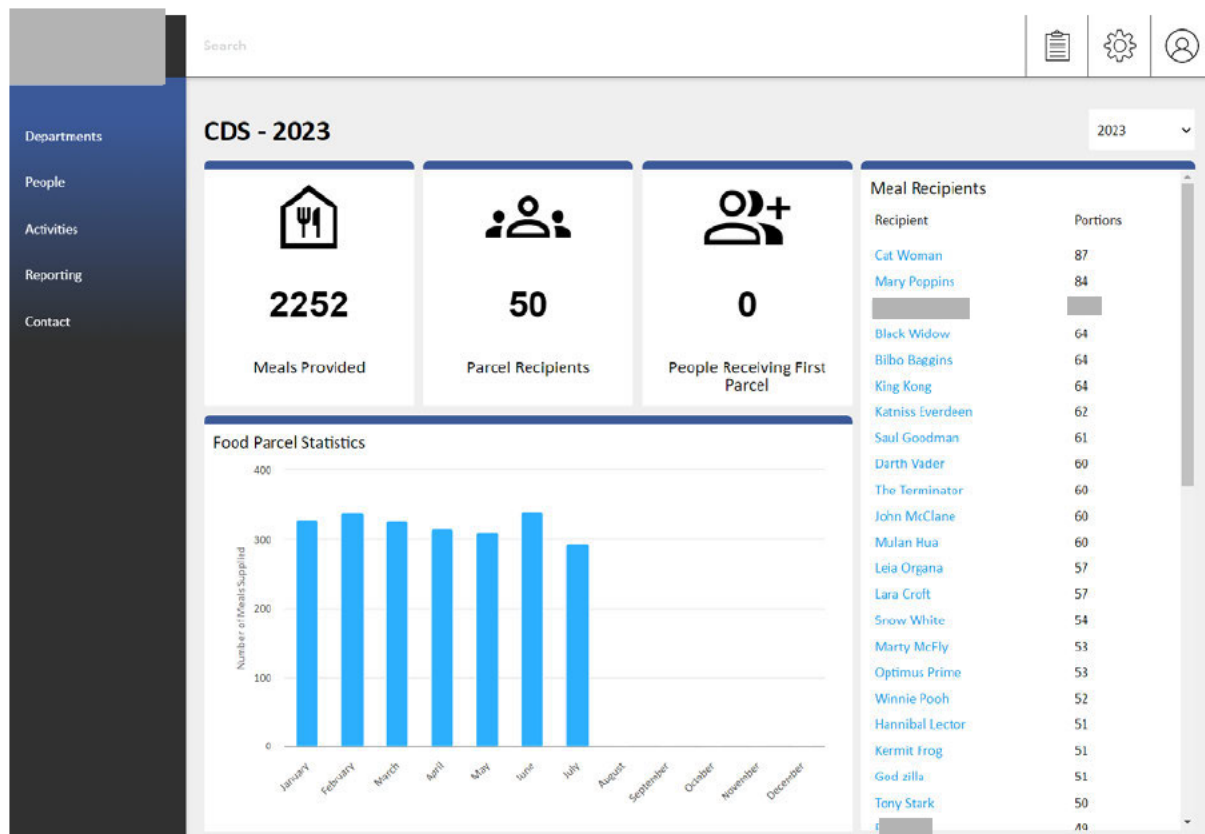


Choosing “All Time” shows overall statistics around meals provided, the number of people that received them, annual totals in a bar graph and a list of people with the portions they received. This gives the big picture of the food bank’s service and impact.



The bars are interactive. Clicking one of the years drills down to a monthly view. The same result is achieved by choosing a year from the time period selector.

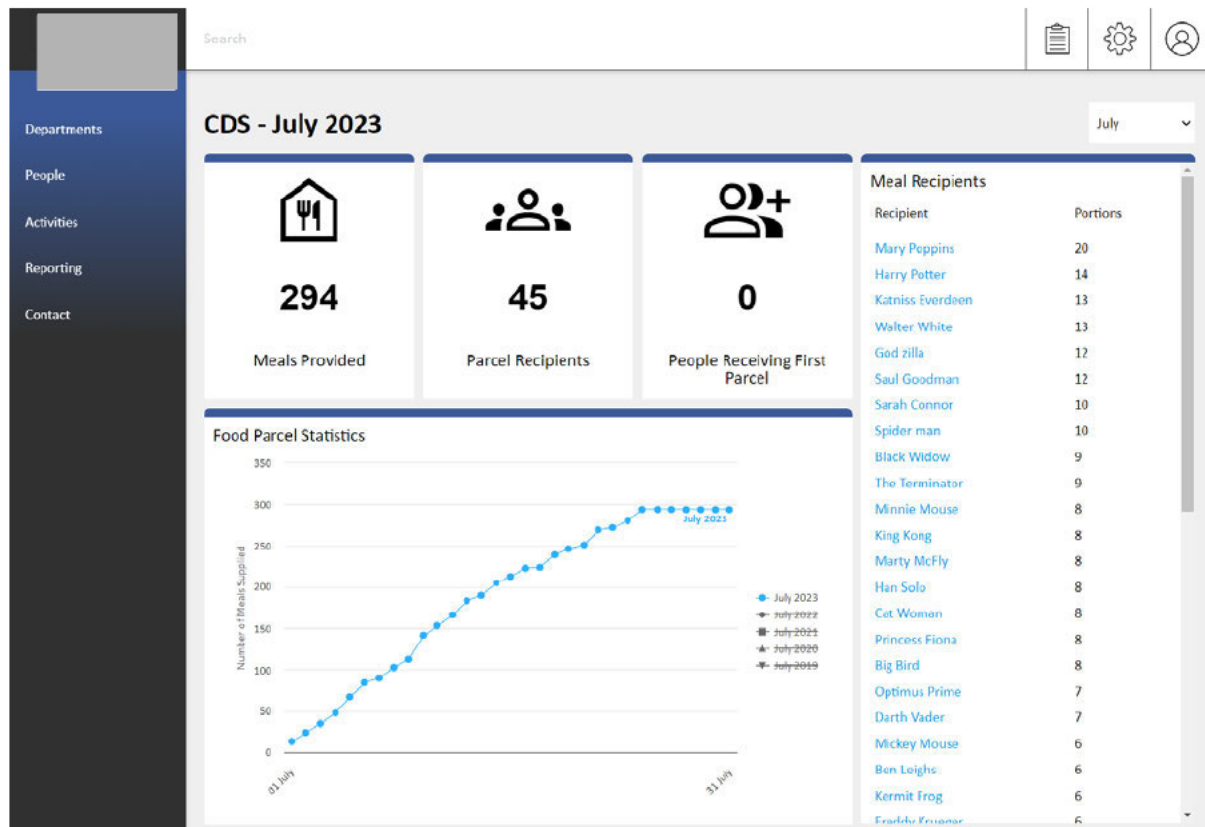
After drilling down to the second level, the statistics and chart focus on a given year. For example, clicking the 2023 bar reloads the page with data for that period:



Both the heading and time selector show 2023, indicating to the user which time period they are looking at. The bar graph shows months along the x-axis instead of years.

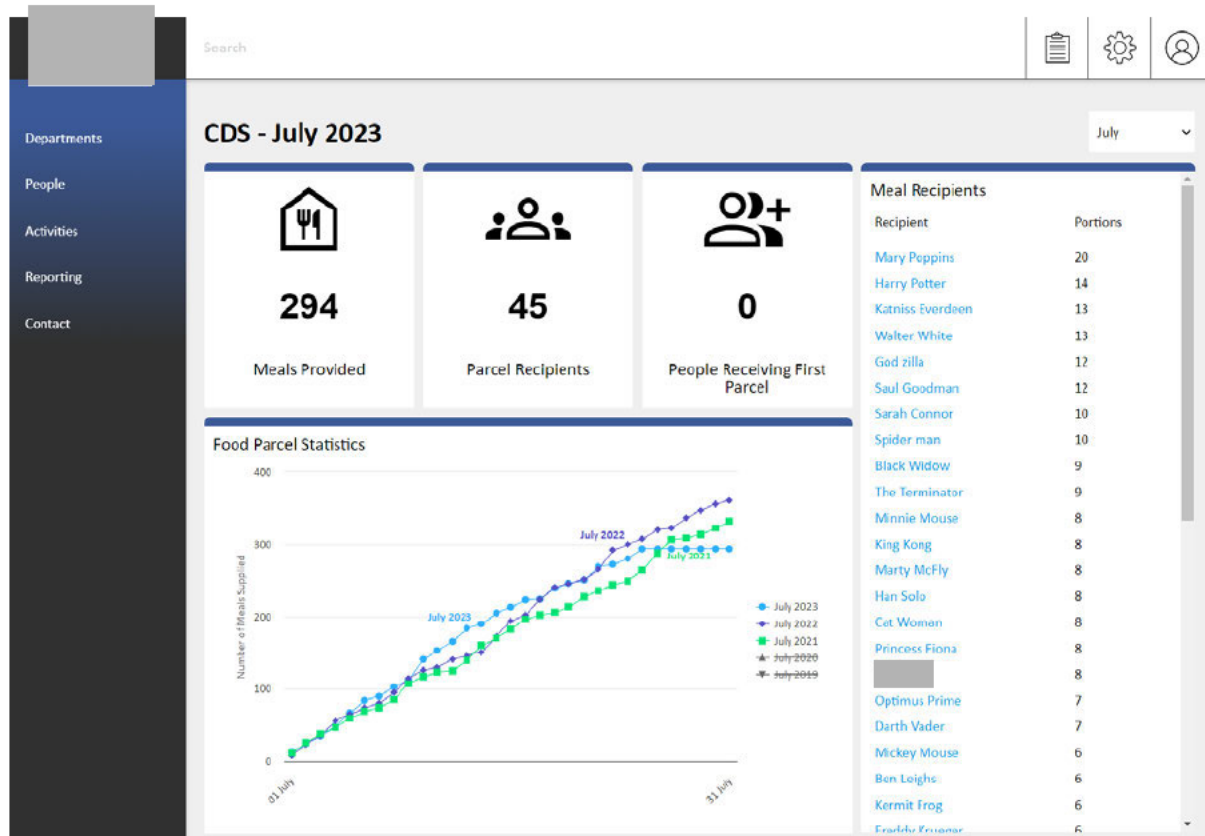
It shows that 2,252 meals were provided in 2023. Note that this is sample data, which is described below. Cat Woman received the most food portions (87 portions in 2023). The bar chart stops in July 2023 as this is when the sample data finished. The columns show that roughly 320 meals were supplied each month in 2023.

Clicking the July column drills down a further level. The page reloads with data from July 2023.



With two clicks, [redacted] can quickly see how many meals were provided in July 2023, which is 294, and that 45 people received parcels.

The bar chart is replaced with a line showing cumulative portions throughout the month. The chart legend shows the same month in prior years. By default prior months are crossed out indicating that the data is not shown on the chart. By clicking them, the cumulative lines from the same month in prior years can be shown, which lets [redacted] compare their impact against prior years.



Clicking July 2022 and July 2021 reveals their lines for comparisons. In this case, July 2022 had the most portions.

Being able to drill down like this is a very powerful tool for [REDACTED]. With a small number of clicks their team can quickly get the statistics that the funders require.

The list of meal recipients is clickable. Clicking on a person shows their details and the dates they received food parcels.

I'm very happy with how the dashboard looks and operates. I went through several design changes before settling on this version. It's visually very pleasing, highly functional and convenient to use.

The dashboard code has three main parts:

- functions relating to the time period
- functions to retrieve data from the database
- functions to display the data as stats, the graphs and table

There is quite a lot of complexity around the time period selector. There are functions to get the period from the URL, work out the start and end dates for SQL statements, work out the name of the period for displaying in the header, and create the drop down time selector dynamically, starting from the current year.

When selecting a time period or clicking on a bar in the chart, a page is loaded with "period=" added to the address bar. For example, "...?period=2023/7" means that the

user chose July 2023 as the period to view. If they chose the whole of 2023, then the address bar would instead have "...?period=2023". I wrote several functions to take the value from the address bar and work out the actual start and end dates whether it's All Time, a whole year or a single month. Function names are:

- year_from_period
- month_from_period
- period_title_description
- start_date_from_period
- end_date_from_period

```
//Look in the address bar and return the year part of the period if available
function year_from_period() {
    $period = get_address_bar_parameter("period");
    if($period=='') {
        return '';
    }
    else if(strpos($period, '/') !== false) {
        //The period contains a forward slash so assume that its year/month
        $array = explode('/', $period);
        return $array[0];
    }
    else {
        return $period;
    }
}

//Look in the address bar and return the month part of the period if available
function month_from_period() {
    $period = get_address_bar_parameter("period");
    //Look for period which is year/month and return the month only
    if(strpos($period, '/') !== false) {
        $array = explode('/', $period);
        return $array[1];
    }
    return 0; //if month isnt found return 0
}

//Converts the period in the address bar into a user readable string
//eg. 2023/7 --> July 2023
function period_title_description() {
    $year = year_from_period();
    $month = month_from_period();
    //If a year isnt selected assume that the user wants "all time"
    if($year=='') {
        return 'All Time';
    }
    //If a month is selected return the month name and the year
    else if($month != 0) {
        return date('F', mktime(0, 0, 0, $month, 1)) . " " . $year;
    }
    //Otherwise assume a year is selected
    else {
        return $year;
    }
}
```

The time period selector is created dynamically with a function "cds_time_drop_list". The function looks at the period from the address bar, and works out what items to

show. It sets the selected list item correctly, so that if the user chooses July then when the page loads the drop list defaults to July.

```
//creates a drop down list which a time parameter can be selected
function cds_time_drop_list() {
    $selected_year = year_from_period();
    $selected_month = month_from_period();
    //add options to choose from the last few years
    $html = "<select id='periodDropList'>";
    $html = $html . "<option value=''>All Time</option>";
    //show the years inside a heading called Yearly
    $html = $html . "<optgroup label='Yearly'>";
    $year = date('Y');
    while($year>2018) {
        $selected_attribute = '';
        if ($year == $selected_year) {
            //if the user choose this year make it selected
            $selected_attribute = ' selected';
        }
        $html = $html . "<option value='$year'$selected_attribute>$year</option>";
        $year = $year - 1;
    }
    $html = $html . "</optgroup>";
    //show the years inside a heading of the selected year
    $html = $html . "<optgroup label='Monthly $selected_year'>";
    //add options to choose a month
    $month = 1;
    while($month<=12) {
        $selected_attribute = '';
        if ($month == $selected_month) {
            //if the user choose this month make it selected
            $selected_attribute = ' selected';
        }
        $month_name = date('F', mktime(0, 0, 0, $month, 1)); //convert month number to name
        $html = $html . "<option value='$selected_year/$month'$selected_attribute>$month_name</option>";
        $month = $month + 1;
    }
    $html = $html . "</optgroup>";
    $html = $html . "</select>";
    return $html;
}
```

The three widgets showing total numbers each have a corresponding function to send a SQL query to the database to get the value:

- read_meal_portion_in_date_range
- read_total_people_with_meal_portion_in_date_range
- read_people_receiving_first_parcel_in_date_range

Each function takes three parameters (the database connection, date from, and date to)

```
//read stats from the database using functions in inc-db-meal.php
$total_meals_provided = read_meal_portion_in_date_range($conn, $from, $to);
$total_people = read_total_people_with_meal_portion_in_date_range($conn, $from, $to);
$new_people = read_people_receiving_first_parcel_in_date_range($conn, $from, $to);
```

The second function returns the number of people that received parcels. It uses count and distinct so that people receiving multiple parcels are only counted as 1 person.

```

//returns the number of people that received a parcel in a set period of time
function read_total_people_with_meal_portion_in_date_range($conn, $from, $to)
{
    //convert the dates into the format that MySQL needs (eg. 2023-01-01)
    $from_string = date('Y-m-d', $from);
    $to_string = date('Y-m-d', $to);
    //create a database query to see how many people have parcels in the date range
    $sql = "select count(distinct PersonId) as Total
            from meal_recipient
            where Date >= '$from_string' and date <= '$to_string'";
    $data = mysqli_query($conn, $sql);
    $row = mysqli_fetch_assoc($data);
    //if no results were found the return 0 parcels
    if ($row["Total"] == null)
    {
        return 0;
    }
    //otherwise return the number of parcels returned from the query result
    return $row["Total"];
}

```

There are three graphs, one showing total meals per year, one showing meals per month of a given year, and the other is the cumulative line graph for a single month.

There is a top-level function that shows a graph based on the time period called "show_food_parcel_chart". Depending on the period, it calls one of three other functions to show the correct graph:

- show_food_parcel_year_bar_chart
- show_food_parcel_month_line_chart
- show_food_parcel_12_month_bar_chart

The first function shows a chart with one bar per year. The function works out the current year and shows 7 bars. It calls a function "read_meal_portion_in_year" for each bar, and build up an array of values for the y-axis and labels for the x-axis. It then uses javascript to create a dictionary in a format that Highcharts expects. The dictionary contains the data points and labels, as well as other information like what to do when clicking a bar. The javascript then instructs Highcharts to show the chart based on the contents of the dictionary.


```

//shows a bar graph where each bar represents one year
function show_food_parcel_year_bar_chart($conn)
{
    //find out the current year and set out how many columns to show
    $current_year = get_current_year();
    $bar_count = 7;
    //create an empty array that will hold the y-axis values and the x-axis labels
    $data_labels = [];
    $data_points = [];
    $year = $current_year - $bar_count + 1;
    while ($year <= $current_year)
    {
        //for each year read the number of portions
        $portions_in_year = read_meal_portion_in_year($conn, $year);
        //add the number of portions and data labels to the arrays
        array_push($data_labels, $year);
        array_push($data_points, $portions_in_year);
        $year = $year + 1;
    }
    //create javascript with a dictionary for Highcharts
    ?>
<script>
window.addEventListener('load', function() {
    renderFoodParcelStatsGraph();
});

//renders the Highcharts graph with the options set
function renderFoodParcelStatsGraph() {
    Highcharts.setOptions( { lang: { decimalPoint: '.', thousandsSep: ',' } } );

    Highcharts.chart('mealGraph', {
        credits: { enabled: false },
        accessibility: { enabled: false },
        title: {text: '', enabled: false},
        chart: {type: 'column'},
        yAxis: {title: {text: 'Number of Meals Supplied'}, labels: { format: '{value:,.0f}' }, softMin: 0, softMax: 10 },
        xAxis: {categories: ["<?php echo (implode("",",", $data_labels)) ?>"]},
        legend: {enabled: false},
        plotOptions: {
            series: {
                cursor: 'pointer',
                label: {connectorAllowed: false},
                point: {
                    events: {
                        click: function () {
                            var url = 'cds.php?period='+this.category;
                            document.location = url;
                        }
                    }
                }
            }
        },
        series: [ { name: 'Year', data: [<?php echo (implode(',', $data_points)) ?>] } ],
        responsive: {
            rules: [{condition: {maxWidth: 500},
                chartOptions: {legend: {layout: 'horizontal', align: 'center', verticalAlign: 'bottom'}}
            }]
        }
    });
}
</script>
<?php
}

```

A similar approach is used for the graph with columns for each month. The third graph with lines is slightly different. It runs 5 queries, one for the month in the current year and prior 4 years. Each data set is added as a series to the javascript dictionary.


```

//create a chart with 5 lines, 4 of which are hidden
function show_food_parcel_month_line_chart($conn)
{
    //work out the period and its dates
    $year = year_from_period();
    $month = month_from_period();
    $month_name = date('F', mktime(0, 0, 0, $month, 1));
    $start_of_month = date("d F", mktime(0, 0, 0, $month, 1, $year));
    $end_of_month = date("d F", mktime(0, 0, 0, $month, cal_days_in_month(CAL_GREGORIAN, $month, $year), $year));
    //read cumulative data for each of the last 5 months
    $data_points_year_minus0 = read_cumulative_meal_portions_for_month($conn, $year - 0, $month);
    $data_points_year_minus1 = read_cumulative_meal_portions_for_month($conn, $year - 1, $month);
    $data_points_year_minus2 = read_cumulative_meal_portions_for_month($conn, $year - 2, $month);
    $data_points_year_minus3 = read_cumulative_meal_portions_for_month($conn, $year - 3, $month);
    $data_points_year_minus4 = read_cumulative_meal_portions_for_month($conn, $year - 4, $month);
    //for the x-axis create a label for the first and last of the month with spaces in between
    $data_labels = [];
    array_push($data_labels, $start_of_month);
    for ($day = 2; $day <= get_days_in_month($year, $month) - 1; $day++)
    {
        array_push($data_labels, '');
    }
    array_push($data_labels, $end_of_month);
    //create javascript with a dictionary for Highcharts
    ?>
    <script>
    window.addEventListener('load', function() {
        renderFoodParcelStatsGraph();
    });

    function renderFoodParcelStatsGraph() {
        Highcharts.chart('mealGraph', {
            credits: { enabled: false },
            accessibility: { enabled: false },
            title: { text: '', enabled: false },
            yAxis: { title: { text: 'Number of Meals Supplied' }, softMin: 0, softMax: 10 },
            xAxis: { categories: ["<?php echo (implode(' ','', $data_labels)) ?>"] },
            legend: { layout: 'vertical', align: 'right', verticalAlign: 'middle' },
            plotOptions: { series: { label: { connectorAllowed: false } } },
            series: [ { name: '<?php echo ($month_name . ' ' . ($year - 0))?>',
                data: [<?php echo (implode(',', $data_points_year_minus0)) ?>],
                visible: true },
                { name: '<?php echo ($month_name . ' ' . ($year - 1))?>',
                data: [<?php echo (implode(',', $data_points_year_minus1)) ?>],
                visible: false },
                { name: '<?php echo ($month_name . ' ' . ($year - 2))?>',
                data: [<?php echo (implode(',', $data_points_year_minus2)) ?>],
                visible: false },
                { name: '<?php echo ($month_name . ' ' . ($year - 3))?>',
                data: [<?php echo (implode(',', $data_points_year_minus3)) ?>],
                visible: false },
                { name: '<?php echo ($month_name . ' ' . ($year - 4))?>',
                data: [<?php echo (implode(',', $data_points_year_minus4)) ?>],
                visible: false }
            ],
            responsive: {
                rules: [{ condition: { maxWidth: 500 },
                    chartOptions: { legend: { layout: 'horizontal', align: 'center', verticalAlign: 'bottom' } }
                }]
            }
        });
    }
    </script>
    <?php
}

```

Sample data used for testing

I wrote PHP code to create random data so that the dashboard is not empty. There are 50 people in the database so far, and the dashboard shows meals over the last 7 years. The PHP code starts with a desired number of food parcels, and creates a SQL INSERT statement for each one. The date of each food parcel uses a random date over the last 7 years, the number of portions is a random number between 1 and 4, and the ID of the recipient person uses a random number between 1 and 50.

To make it more realistic, I used a random number function that distributes the dates more towards the current date. This gives the effect that the food bank is growing, and causes the yearly bar graph to get bigger each year.

The PHP code looks like this:

```
<!DOCTYPE html>
<html>
<head>
  <title>Random Food Parcel Data</title>
</head>

<body>
<h1>Random Food Parcel Data</h1>
<pre>
<?php

//return a random number between in a range, which is distributed more at the upper end
function skewed_random_number($min, $max, $power_factor)
{
    $range = $max - $min + 1;
    $rand1 = mt_rand() / mt_getrandmax();
    $rand2 = mt_rand() / mt_getrandmax();
    $skewedRand = $min + pow($rand1, $power_factor) * ($max - $min + 1);
    $skewedRand = min(max($skewedRand, $min), $max);
    return (int)$skewedRand;
}

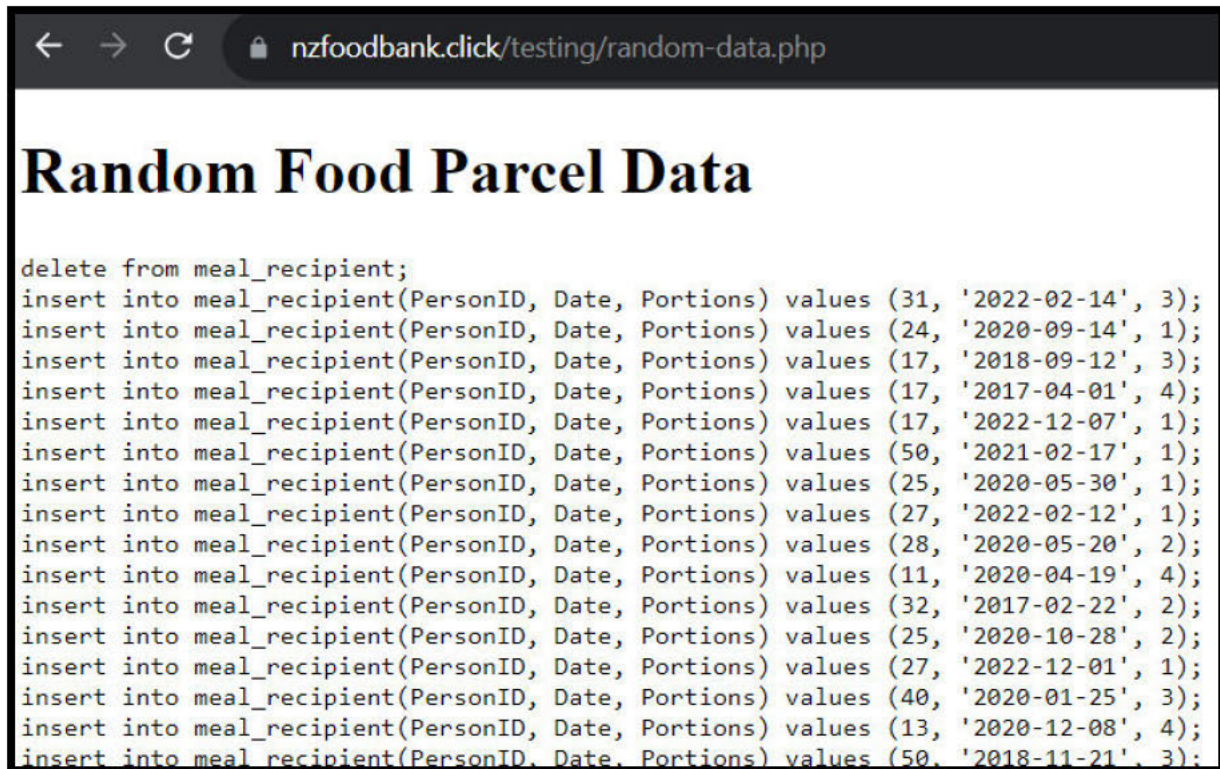
//configuration options here
$people_count = 50;
$parcel_count = 11632;
$from = new DateTimeImmutable('5 May 2016');
$to = new DateTimeImmutable('25 Jul 2023');

//random data creation starts here
$day_count = $from->diff($to)->days;
echo "delete from meal_recipient;\n";
for ($index = 0; $index < $parcel_count; $index++)
{
    $random_number_of_days = skewed_random_number(0, $day_count, 0.8);
    $random_person_id = skewed_random_number(1, $people_count, 0.8);
    $random_portions = 5 - skewed_random_number(1, 4, 0.6);

    $random_date = $from->modify(round($random_number_of_days) . ' days'); //adds the random number of days to the "from" date
    $random_date_as_string = $random_date->format('Y-m-d');
    $sql = "insert into meal_recipient(PersonID, Date, Portions) values ($random_person_id, '$random_date_as_string', $random_portions);\n";
    echo $sql;
}

?>
</pre>
</body>
</html>
```

The output is a web page with 11,632 SQL statements.



```

delete from meal_recipient;
insert into meal_recipient(PersonID, Date, Portions) values (31, '2022-02-14', 3);
insert into meal_recipient(PersonID, Date, Portions) values (24, '2020-09-14', 1);
insert into meal_recipient(PersonID, Date, Portions) values (17, '2018-09-12', 3);
insert into meal_recipient(PersonID, Date, Portions) values (17, '2017-04-01', 4);
insert into meal_recipient(PersonID, Date, Portions) values (17, '2022-12-07', 1);
insert into meal_recipient(PersonID, Date, Portions) values (50, '2021-02-17', 1);
insert into meal_recipient(PersonID, Date, Portions) values (25, '2020-05-30', 1);
insert into meal_recipient(PersonID, Date, Portions) values (27, '2022-02-12', 1);
insert into meal_recipient(PersonID, Date, Portions) values (28, '2020-05-20', 2);
insert into meal_recipient(PersonID, Date, Portions) values (11, '2020-04-19', 4);
insert into meal_recipient(PersonID, Date, Portions) values (32, '2017-02-22', 2);
insert into meal_recipient(PersonID, Date, Portions) values (25, '2020-10-28', 2);
insert into meal_recipient(PersonID, Date, Portions) values (27, '2022-12-01', 1);
insert into meal_recipient(PersonID, Date, Portions) values (40, '2020-01-25', 3);
insert into meal_recipient(PersonID, Date, Portions) values (13, '2020-12-08', 4);
insert into meal_recipient(PersonID, Date, Portions) values (50, '2018-11-21', 3);

```

I copied the SQL to a file, uploaded it to my server and ran it on the live database. The result is that the dashboard shows meaningful data for testing, and also a realistic view of the system for [redacted] to see.

Testing phase 2

Feature and Type	Expected Outcome	Actual Outcome	Action
CDS dashboard			
Statistics widgets Expected case	The number of meals provided should match what's in the database	Both the database query and dashboard show the correct number which is 23,531	None needed
Chart consistency Expected case	When drilling down through the charts, the sum of data should match the parent chart's bar value	The July 2023 bar shows 2,252 meals, which matches the sum of bars of the monthly 2023 chart.	None needed
Future period selection Boundary case	The stats widgets and charts should show zeroes and no errors should appear.	The dashboard shows zeroes and the charts are empty.	None needed

Feature and Type	Expected Outcome	Actual Outcome	Action
Invalid case Invalid period selection	The dashboard should default to "All Time" if the period in the address bar is invalid or missing.	PHP error DateTime setDate should be of type int appears when setting the period in the address bar to "hello"	The PHP code needs to check the period format, and reject it if the month or year are not numbers. Debugged and fixed on 15 Sep 2023
Viewing a person			
Invalid case Missing person ID in the address bar	A friendly message explains that the person does not exist	Multiple PHP errors	The PHP code needs to check that the person ID is a valid integer, and if not show a message to the user Debugged and fixed on 15 Sep 2023
Editing a person			
Invalid case Name validation	Saving a person without specifying a name should show a message to the user and prevent the save	A popup alert "Please type a first name" is shown.	None needed
Expected case Selected departments saved to the database	Various combinations of department tick boxes should be saved correctly when editing a person.	Viewing a person after saving shows the correct departments.	None needed
Boundary case Entering a date of birth beyond the current date should be prevented	The popup calendar should not allow dates in the future.	Dates in the following year are not selectable	None needed
Invalid case	Date of birth should only be	The popup calendar means that the user	None needed

Feature and Type	Expected Outcome	Actual Outcome	Action
Date of birth date formatting	accepted in a valid DMY format	doesn't type dates, and therefore always use the correct format	
Expected case Address auto-complete	Searching for an address should give matches and selecting one should fill in the Street, Suburb, City and Post Code fields	Works as expected	None needed
Adding a food parcel			
Number of portions less than zero Boundary case	Typing 0 or a negative number of portions should show the user an alert	The message "Please enter a positive number of portions"	None needed
Expected case New meal portions appear in dashboard stats and chart	New food parcels added for a person should appear in the dashboard stats	The widgets showing meals provided, and the charts increased by the number of portions, as expected	None needed
Expected case New meal portions should appear in the person's history	Newly added food parcels should be listed when viewing a person	The list of meals updates successfully, showing the new food parcel and date	None needed
Other functionality			
Expected case Searching for a person should include name, email and phone	Searching for batman should give a single result, and searching for 022 should find a matching phone number.	Search results were found by searching for names, phone numbers and email addresses	None needed
Invalid case Invalid or missing user login and password	Logging in without a correct username and password should	A message "You are not authorised to view this page" is shown.	None needed

Feature and Type	Expected Outcome	Actual Outcome	Action
	deny access to the system.		

Reflection of project management tool

ClickUp was a vital tool in this project, I already had issues with time management and without my project management tool it would've been much worse. However if I did everything all over again I probably would have looked harder to find something better. It was overly complicated for my situation with a huge amount of features that were not relevant to me at all. I was lost when using it to start off with which wasted a bit of time however I am now daily comfortable with it. For a tool that is supposed to save me time (which it did) I still spent a lot of time figuring out how to use it. Perhaps one of the other project management tools I researched would've been a better option. Nevertheless ClickUp is still an impressive tool that did help me stay on track throughout the entire project.

Presenting the system

Instead of sending the link to the project or a video of me using it I wanted to deliver the project to [redacted] a personally and show them its features. That way I can get their direct thoughts and avoid any miscommunication of the system's features by being able to walk them through it.

These are my notes from the last meeting with [redacted], my contact at [redacted]. This is my final stakeholder feedback for the school project.

from [redacted]'s comments

- [redacted] is happy with the address finder and sees that it will make it quicker, easier and more accurate to set up new clients. She thinks this feature is perfect and doesn't require any changes
- They liked the date selector as it is nicer to use than typing dates. [redacted] suggested that it would be useful to have a date range selector on the dashboard to get reporting between precise dates
- They're very happy with the style and professional look of the system. It's a big upgrade from their last system, where usability and aesthetics weren't considerations
- [redacted] said that the home page is exactly how we discussed, and she liked that her design ended up in the final product. Dividing the home page by department matches up with staff roles at [redacted] making it a good fit.
- [redacted] appreciated that her description of the dashboard with totals and the chart closely resembles the final product. The ability to compare historic months on a single chart was a nice surprise for them.

Opportunities for improvement and next steps

- It would be helpful for [redacted] if the dashboard could download reports as PDFs or spreadsheets rather than copying and pasting from the screen.
- More precise time period control on the dashboard would be helpful, for example being able to report by quarters and half years rather than just full years and months

- Adding activities like sessions attended is the ideal next step, with the dashboards for Evergreen Club and Friendship Link following.

Future plans

My plan is to continue working with [REDACTED] to finalise the product and make it fully functional. From the start of this year, I knew that [REDACTED]'s needs were more complicated and time consuming than I could do for the school project. To deal with this, we agreed to focus on the foodbank part of the system, and leave the Evergreen Club and Friendship Link for 2024. I see the relationship carrying on for many years, with lots of improvements and new features coming over time.

The foodbank part of the system would be relevant to lots of different food banks in New Zealand, and potentially the world. I would love to see my system develop in a direction that gets it being used by lots of different organisations throughout New Zealand. To make that happen, I would need a marketing website to promote the system, and contact information of food banks in NZ. This is an exciting possibility, and I intend to follow through with it.

By supporting food banks, I can make their operations run smoothly with my system, and have a positive impact on lots of people that use their services.

I am very happy and grateful for the opportunity to create a project for a real world customer. It has been a great experience and I have learned a lot along the way. Now that I have a marketable product, I am considering turning it into a commercial product. Not in a financial sense, but rather to help people in need. I am gaining skills and experience that align with my career aspirations as a software designer.

Video demonstration of the system

Here is a link to a full presentation of my system and all of its current features. Please note there are still many features unfinished which is why the video does not go into every page or button. I tried multiple recording applications but unfortunately none seemed to record the drop down lists for the gender or title fields properly.



Scholarship

Subject: Technology

Standard: 93601

Total score: 15

Q	Score	Marker commentary
Synthesis and integration	06	<p>This project developed an outcome for an authentic issue of creating a database for a food bank that enabled stakeholders to maintain records and run reports.</p> <p>The final outcome was well designed, utilising a range of techniques to develop a website that was aesthetic, functional, and easy to use.</p> <p>The database design supported all functionality and future growth.</p> <p>There was deliberate decision-making around the software used and the techniques and processes employed during development.</p> <p>Authentic stakeholder involvement enabled effective decision making.</p> <p>The outcome produced clearly met the brief requirements, with these featuring in the commentary throughout the project.</p> <p>There was evidence of ongoing testing and evaluation throughout. The website was hosted online, enabling in-situ testing by stakeholders.</p> <p>The technological practice utilised, and the outcome produced demonstrate some elegant synthesis of knowledge, skills, ideas, and methods to drive and complete the project.</p>
Justification	05	<p>Evidence of systematic development. Markers were confidently able to see the process and understand why decisions were made.</p> <p>Stakeholders were vital to the development process. Because they were genuinely invested in the outcome, their feedback was ongoing, authentic, and meaningful and was reflected in the final outcome.</p> <p>Throughout the process, clear next steps and improvements were identified. Analysis throughout the report was relevant and purposeful.</p> <p>There is evidence of functional modelling, including relevant testing and evaluation of techniques / processes / mockups which has enabled informed decisions during project development.</p> <p>Both subjective and objective tests were explained and utilized appropriately, and as a result of this testing,</p>

		<p>markers felt confident that the outcome was going to be realised in due process.</p> <p>The reports that were generated were made with the end user requirements in mind. They were easy to access and interpret.</p> <p>Feedback was sought from stakeholders, which justified its fitness for purpose in the broadest sense.</p>
Critical reflection	04	<p>The report demonstrated a clear narrative from start to finish. The candidate demonstrated reflection on key decisions throughout.</p> <p>There was consistent, clear evidence of the candidate's thinking about the process and outcome.</p> <p>There is a reasonable level of relevant reflection and evaluation evident throughout that was always clear. This enabled the candidate to make informed and reasoned decisions, modifications, and adaptations during development as issues arose. At the end of the project, the candidate reflected on the outcome's overall success and identified an area for further improvement investigation.</p> <p>The markers were able to understand why the candidate was making the decisions they made.</p> <p>There was evidence that candidate reflection informed their decision-making and subsequent selection of tools and techniques.</p> <p>The end result was a professional website that is functional and ready to be used by the food bank.</p>